

# Proiect - Sisteme de Gestiune a Bazelor de Date

- Gestiunea unui lanț de restaurante -

Popescu Paullo Robertto Karloss

Grupa 231

2022 Ianuarie

# Cuprins:

1. Prezentarea bazei de date.
  - 1.1 Tehnologii folosite pentru realizarea proiectului.
  - 1.2 Descrierea temei alese.
  - 1.3 Prezentarea constrângerilor impuse asupra modelului.
  - 1.4 Descrierea entităților.
  - 1.5 Descrierea relațiilor.
  - 1.6 Descrierea atributelor.
2. Diagrama Entitate-Relație (ER).
3. Diagrama Conceptuală.
4. Schemele relaționale corespunzătoare diagramei conceptuale.
5. Implementarea bazei de date în Oracle.
  - 5.1 Crearea tabelelor și a constrângerilor.
  - 5.2 Inserarea datelor coerente în tabele (minimum 5 înregistrări în fiecare tabel neasociativ, minimum 10 înregistrări în tabelele asociative) + crearea unei secvențe.
  - 5.3 Diagrama generată în sql după crearea tabelelor și inserarea datelor.
6. Crearea unui *subprogram stocat* care utilizează două tipuri de colecție studiate.
7. Crearea unui *subprogram stocat* care utilizează un tip de cursor studiat.
8. Crearea unui *subprogram stocat de tip funcție*, care utilizează într-o singură comandă SQL minim 3 tabele (tratarea și punerea în evidență a excepțiilor).
9. Crearea unui *subprogram stocat de tip procedură*, care utilizează într-o singură comandă SQL minim 5 tabele (tratarea și punerea în evidență a excepțiilor, NO\_DATA\_FOUND și TOO\_MANY\_ROWS).
10. Crearea unui Trigger LMD la nivel de comandă.
  - 10.1 Declanșarea acestuia.
11. Crearea unui Trigger LMD la nivel de linie.
  - 11.1 Declanșarea acestuia.
12. Crearea unui Trigger LDD.
  - 12.1 Declanșarea acestuia.
13. Crearea unui pachet care conține obiectele de la punctele 6-9.
14. Crearea unui pachet care conține tipuri de date complexe și obiecte necesarea unui flux de acțiuni integrate, specifice bazei de date.

## **1. Prezentarea bazei de date**

### ***1.1 Tehnologii folosite pentru realizarea proiectului.***

Pentru proiectul din cadrul cursului de Baze de Date am folosit versiunea **o19c** a *Oracle Database*.

Ce aplicații am folosit?

- Oracle SQL Developer

### ***1.2 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 usoară ținerea în evidență a mai multor restaurante (comenzi, angajați, facturi etc).

### ***1.3 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.

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, şampanie etc).

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

### ***1.4 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 postal 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 comenzi*.
- Entitatea **Angajat**, care va conține 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. Cheia primară fiind id-ul angajatului.
- Subentitatea **Chelner**, care va conține numărul numărul de ani de experiență al fiecărui chelner. Cheia primară fiind id-ul angajatului.
- Subentitatea **Manager**, care nu are attribute, 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 nevoie 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*.

### **1.5 Descrierea relațiilor.**

Un anumit produs poate apartine mai multor restaurante.

Într-o locație poate fi găsit un singur restaurant.

Un produs se poate regăsi în mai multe comenzi.

Într-o comandă se pot găsi 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.

### 1.6 Descrierea atributelor.

- Entitatea **restaurant** va avea ca atrbute 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 atrbute: 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 atrbute: 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 atrbute: 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 atrbute: 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 atrbute: 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 atrbute: 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 atrbute: 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 comenzzii 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 comenzzii (cheia primară).

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

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

- Tabelul asociativ **conținut\_comandă** va avea ca atrbute: id-ul produsului, id-ul comenzzii și numărul de produse.

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

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

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

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

Angajat va avea ca cheie externă id-restaurant (provenită din tabelul restaurant).

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

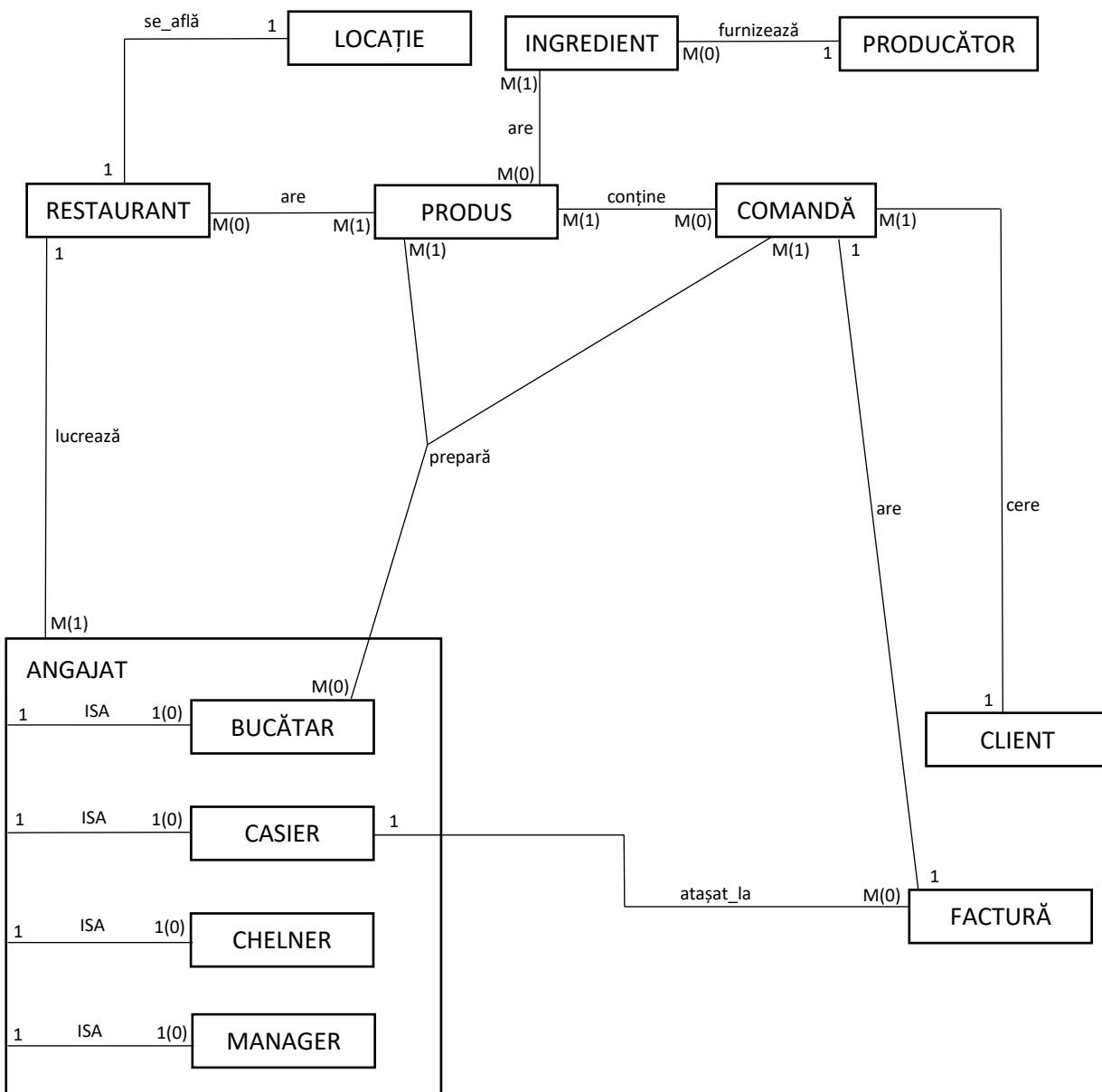
- Subentitatea **casier** va avea ca atrbute id-ul angajatului și numărul de ani de studii.

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

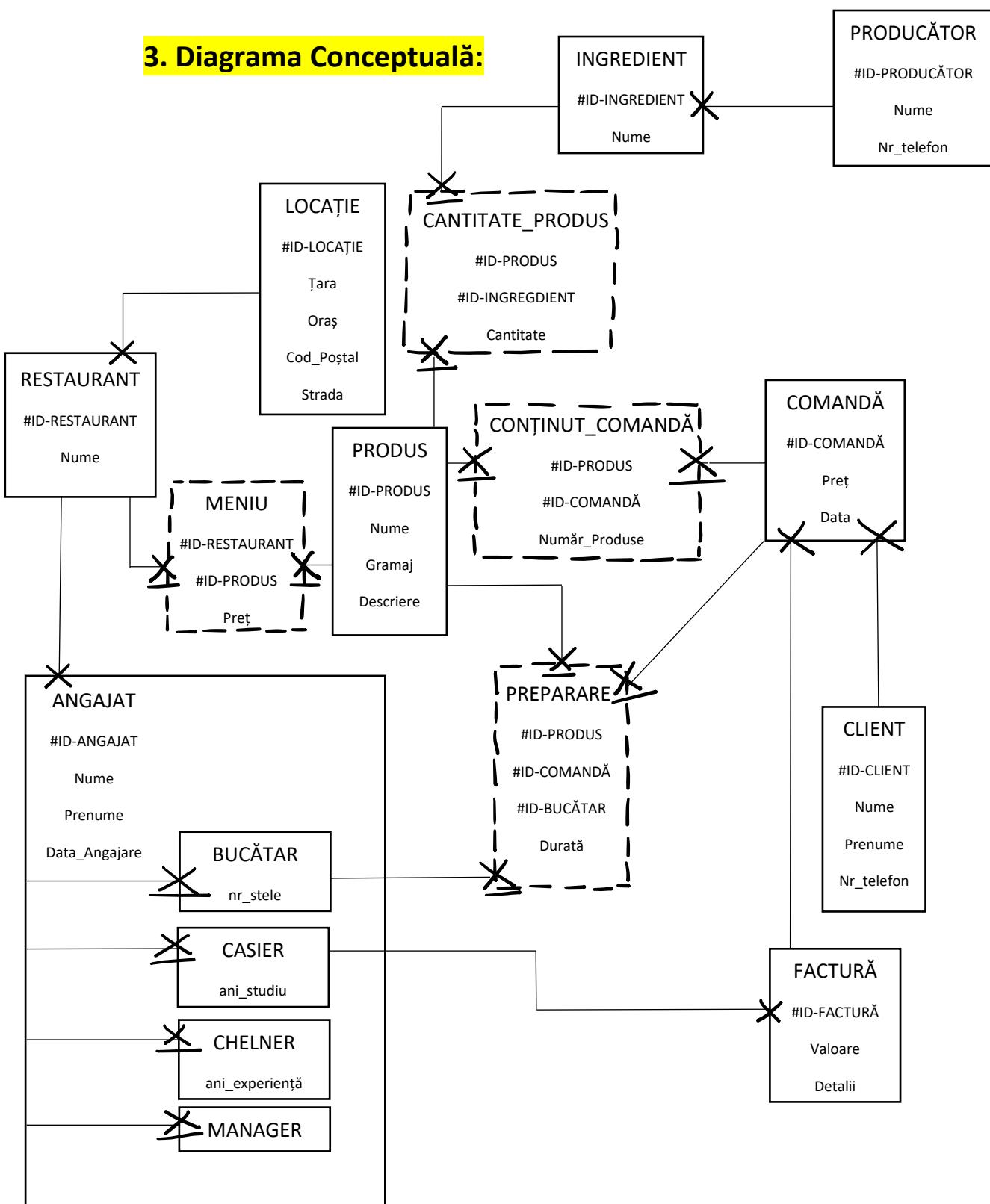
PK va fi id\_angajat (care provine din tabelul angajat).

- Subentitatea **chelner** va avea ca atrbute 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 atrbut id-ul angajatului. 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 atrbute: id-ul produsului, id-ul comenzi, id-ul bucătarului și durata.  
Avem o cheie primară compusă formată din id-ul produsului (FK din tabelul produs), id-ul comenzi (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 comenzi number(20), id-ul bucătarului number(10) și durata number(10).
- Entitatea **client** va avea ca atrbute: 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 atrbute: 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):



### 3. Diagrama Conceptuală:



#### **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-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. Implementarea bazei de date în Oracle:**

##### **5.1 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:**

The screenshot shows the 'Columns' tab of the LOCATIE table in Oracle SQL Developer. The table has five columns: ID, LOCATIE, TARA, ORAS, and COD\_POSTAL. The ID column is a primary key (NUMBER(10,0)) and is not nullable. The other four columns are VARCHAR2(20 BYTE) and are nullable.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID	LOCATIE NUMBER(10,0)	No	(null)	1	(null)
2 TARA	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)
3 ORAS	VARCHAR2(20 BYTE)	Yes	(null)	3	(null)
4 COD_POSTAL	VARCHAR2(15 BYTE)	Yes	(null)	4	(null)
5 STRADA	VARCHAR2(50 BYTE)	Yes	(null)	5	(null)

-- 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:**

The screenshot shows the 'Columns' tab of the RESTAURANT table in Oracle SQL Developer. The table has three columns: ID, RESTAURANT, and NUME. The ID column is a primary key (NUMBER(10,0)) and is not nullable. The RESTAURANT column is NUMBER(10,0) and is nullable. The NUME column is VARCHAR2(32 BYTE) and is nullable.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID	RESTAURANT NUMBER(10,0)	No	(null)	1	(null)
2 ID_LOCATIE	NUMBER(10,0)	Yes	(null)	2	(null)
3 NUME	VARCHAR2(32 BYTE)	Yes	(null)	3	(null)

-- 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:***

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID	PRODUS NUMBER (10, 0)	No	(null)	1	(null)
2 NUME	VARCHAR2(25 BYTE)	Yes	(null)	2	(null)
3 GRAMAJ	NUMBER (10, 0)	Yes	(null)	3	(null)
4 DESCRIERE	VARCHAR2(100 BYTE)	Yes	(null)	4	(null)

-- MENUU --

```

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:**

The screenshot shows the 'Columns' tab in Oracle SQL Developer for the 'PROJECT BD ROBERTTO1.sql' schema. The table name is 'PRODUS'. The columns listed are: ID (NUMBER(10,0)), PRODUS (NUMBER(10,0)), and PRET (NUMBER(10,0)). The 'ID' column is defined as the primary key (PRIMARY KEY).

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID	NUMBER (10, 0)	No	(null)	1	(null)
2 ID RESTAURANT	NUMBER (10, 0)	No	(null)	2	(null)
3 PRET	NUMBER (10, 0)	Yes	(null)	3	(null)

-- PRODUCATOR --

```
CREATE TABLE producator (
    id_producator NUMBER(10) PRIMARY KEY,
    nume        VARCHAR2(32),
    nr_telefon  VARCHAR2(15) UNIQUE
);
```

**Print-Screen:**

The screenshot shows the 'Columns' tab in Oracle SQL Developer for the 'PROJECT BD ROBERTTO1.sql' schema. The table name is 'PRODUCATOR'. The columns listed are: ID (NUMBER(10,0)), NUME (VARCHAR2(32)), and NR TELEFON (VARCHAR2(15)). The 'ID' column is defined as the primary key (PRIMARY KEY).

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID	NUMBER (10, 0)	No	(null)	1	(null)
2 NUME	VARCHAR2 (32 BYTE)	Yes	(null)	2	(null)
3 NR TELEFON	VARCHAR2 (15 BYTE)	Yes	(null)	3	(null)

-- 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:**

The screenshot shows the 'PROJECT BD ROBERTTO1.sql' tab selected in the top bar. Below it, the 'INGREDIENT' table is displayed in a grid format. The columns are: COLUMN\_NAME, DATA\_TYPE, NULLABLE, DATA\_DEFAULT, COLUMN\_ID, and COMMENTS. The data rows are:

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID	NUMBER(12,0)	No	(null)	1	(null)
2 ID	NUMBER(10,0)	Yes	(null)	2	(null)
3 NUME	VARCHAR2(32 BYTE)	Yes	(null)	3	(null)

-- 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:**

The screenshot shows the 'PROJECT BD ROBERTTO1.sql' tab selected in the top bar. Below it, the 'CANTITATE\_PRODUS' table is displayed in a grid format. The columns are: COLUMN\_NAME, DATA\_TYPE, NULLABLE, DATA\_DEFAULT, COLUMN\_ID, and COMMENTS. The data rows are:

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID	NUMBER(10,0)	No	(null)	1	(null)
2 ID	NUMBER(12,0)	No	(null)	2	(null)
3 CANTITATE	NUMBER(10,0)	Yes	(null)	3	(null)

-- ANGAJAT --

```
CREATE TABLE angajat (
```

```

id_angajat NUMBER(10) PRIMARY KEY,
id_restaurant 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 )

);

```

**Print-Screen:**

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID_ANGAJAT	NUMBER(10,0)	No	(null)	1	(null)
2 ID_RESTAURANT	NUMBER(10,0)	Yes	(null)	2	(null)
3 NUME	VARCHAR2(32 BYTE)	Yes	(null)	3	(null)
4 PRENUME	VARCHAR2(32 BYTE)	Yes	(null)	4	(null)
5 DATA_ANGAJARE	DATE	Yes	to_date(sysdate, 'dd-mm-yy')	5	(null)

-- 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:**

The screenshot shows the Oracle SQL Developer interface with the 'CHELNER' tab selected. The 'Columns' tab is active, displaying the following table structure:

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID ANGAJAT	NUMBER (10, 0)	No	(null)	1	(null)
2 ANI EXPERIENTA	NUMBER (5, 0)	Yes	(null)	2	(null)

-- 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:**

The screenshot shows the Oracle SQL Developer interface with the 'CASIER' tab selected. The 'Columns' tab is active, displaying the following table structure:

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID ANGAJAT	NUMBER (10, 0)	No	(null)	1	(null)
2 ANI STUDIU	NUMBER (5, 0)	Yes	(null)	2	(null)

-- 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:**

The screenshot shows the 'Columns' tab for the 'BUCATAR' table in the 'PROJECT BD ROBERTTO1.sql' schema. The table has two columns: 'ID' and 'NR'. The 'ID' column is of type NUMBER(10,0) and is the primary key (PK). The 'NR' column is of type NUMBER(7,0) and is a foreign key (FK) referencing the 'id\_angajat' column in the 'ANGAJAT' table.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID	ANGAJAT NUMBER(10, 0)	No	(null)	1	(null)
2 NR	STELE NUMBER(7, 0)	Yes	(null)	2	(null)

-- 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:**

The screenshot shows the 'Columns' tab for the 'MANAGER' table in the 'PROJECT BD ROBERTTO1.sql' schema. The table has one column: 'ID'. The 'ID' column is of type NUMBER(10,0) and is the primary key (PK).

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID	ANGAJAT NUMBER(10, 0)	No	(null)	1	(null)

-- 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:***

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID	FACTURA NUMBER (15, 0)	No	(null)	1	(null)
2 ID_ANGAJAT	NUMBER (10, 0)	Yes	(null)	2	(null)
3 VALOARE	NUMBER (10, 0)	Yes	(null)	3	(null)
4 DETALII	VARCHAR2 (50 BYTE)	Yes	(null)	4	(null)

-- 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:***

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID_CLIENT	NUMBER (10, 0)	No	(null)	1	(null)
2 NUME	VARCHAR2 (32 BYTE)	No	(null)	2	(null)
3 PRENUME	VARCHAR2 (32 BYTE)	No	(null)	3	(null)
4 NR_TELEFON	VARCHAR2 (15 BYTE)	Yes	(null)	4	(null)

-- 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:***

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID	COMANDA NUMBER (20, 0)	No	(null)	1	(null)
2 ID_CLIENT	NUMBER (10, 0)	Yes	(null)	2	(null)
3 ID_FACTURA	NUMBER (15, 0)	Yes	(null)	3	(null)
4 PRET	NUMBER (10, 0)	No	(null)	4	(null)
5 DATA	DATE	Yes	to date(sysdate, 'dd-mm-yy')	5	(null)

-- 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:***

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID_PRODUS	NUMBER(10,0)	No	(null)	1	(null)
2 ID_COMANDA	NUMBER(20,0)	No	(null)	2	(null)
3 NUMAR_PRODUSE	NUMBER(10,0)	Yes	(null)	3	(null)

```
-- 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 )
);

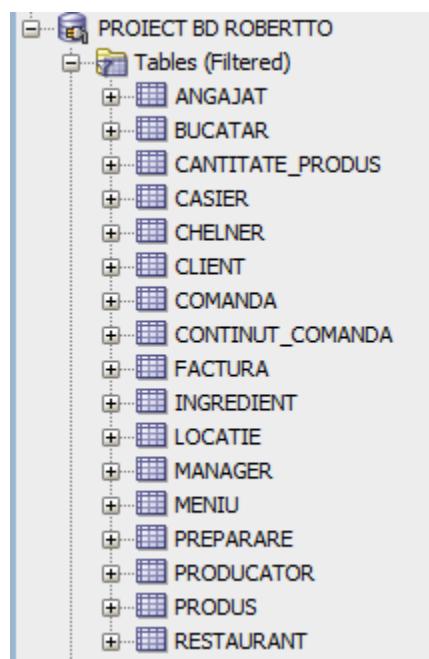
```

**Print-Screen:**

The screenshot shows the 'Columns' tab of the PRODUS table in the PROJECT BD ROBERTTO1.sql schema. The table has four columns: ID, PRODUS, DATA\_TYPE, NULLABLE, DATA\_DEFAULT, COLUMN\_ID, and COMMENTS. The data is as follows:

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID	PRODUS NUMBER (10, 0)	No	(null)	1	(null)
2 ID	COMANDA NUMBER (20, 0)	No	(null)	2	(null)
3 ID	ANGAJAT NUMBER (10, 0)	No	(null)	3	(null)
4 DURATA	NUMBER (10, 0)	Yes	(null)	4	(null)

**Print-Screen:**



**5.2 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:**

A screenshot of a SQL query result window. The query is "select \* from locatie;". The results show five rows of data:

ID_LOCATIE	TARA	ORAS	COD_POSTAL	STRADA
1	Romania	Bucuresti	010051	Batista
2	Romania	Cluj	400033	Mihai Eminescu
3	Romania	Iasi	700547	Rediu
4	Italia	Milano	20127	Monza
5	Germania	Munchen	80687	Perhamerstrabe

```
-- 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:**

The screenshot shows a MySQL query editor window. The SQL query entered is: `249 | select * from restaurant;`. Below the query, the results are displayed in a table titled "Query Result". The table has three columns: ID\_RESTAURANT, ID\_LOCATIE, and NUME. The data is as follows:

ID_RESTAURANT	ID_LOCATIE	NUME
1	1	2 Gurmandul
2	2	3 Yamas
3	3	1 Ivans
4	4	5 Savanna
5	5	4 Grande appetito!

```
-- 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:***

The screenshot shows a SQL query interface with a toolbar at the top and two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab is active, displaying the query 'select \* from meniu;'. Below the query is a 'Query Result' table with 11 rows of data. The table has three columns: ID\_PRODUS, ID\_RESTAURANT, and PRET. The data is as follows:

ID_PRODUS	ID_RESTAURANT	PRET
1	1	35
2	2	50
3	5	20
4	1	33
5	2	60
6	4	15
7	4	13
8	5	12
9	4	30
10	4	15
11	3	27

```
-- 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:***

The screenshot shows a database interface with a toolbar at the top and two main panes below. The top pane is labeled 'Worksheet' and contains a SQL query: 'select \* from produs;'. The bottom pane is labeled 'Query Result' and displays the results of the query in a table format.

ID_PRODUS	NUME	GRAMAJ	DESCRIERE
1	1 Spaghete	200	picante
2	2 Pizza	150	dulce aromata
3	3 Paste boloanese	200	cu sos dulce
4	4 Sarmale	250	ca la mama acasa
5	5 Ciorba de post	300	fara carne
6	6 Sushi	300	(null)

```
-- 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:***

The screenshot shows the Oracle SQL Developer interface. In the top-left, there's a code editor window with the number '551' and the SQL command: 'select \* from cantitate\_produs;'. Below it is a toolbar with icons for script, output, and query result. The main area is titled 'Query Result' and shows a table with 12 rows of data. The table has three columns: ID\_PRODUS, ID\_INGREDIENT, and CANTITATE. The data is as follows:

ID_PRODUS	ID_INGREDIENT	CANTITATE
1	1	85
2	1	50
3	2	60
4	2	150
5	2	100
6	3	200
7	3	55
8	4	100
9	4	500
10	5	350
11	2	50
12	2	35

```
-- 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:**

511 | `select * from ingredient;`

Script Output x Query Result x

SQL | All Rows Fetched: 7 in 0.003 seconds

ID_INGREDIENT	ID_PRODUCATOR	NUME
1	1	1 rosii
2	2	3 varza murata
3	3	4 ulei floarea soarelui
4	4	5 condimente
5	5	2 bors
6	6	2 parmezan
7	7	2 cascaval

-- 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:**

The screenshot shows the Oracle SQL Developer interface. In the top-left corner, there's a toolbar with various icons. Below it, the main window has tabs for 'Worksheet' and 'Query Builder'. The 'Worksheet' tab is active, displaying the SQL query: 'select \* from producator;'. The results are shown in a table titled 'Query Result' with 7 rows. The table has three columns: 'ID\_PRODUCATOR', 'NUME', and 'NR\_TELEFON'. The data is as follows:

ID_PRODUCATOR	NUME	NR_TELEFON
1	Livada cu de toate	0770573182
2	Olivers	0754754318
3	FreshOnly	021999123
4	ION MOS	021129123
5	FUCHS	021592555
6	Bergenbier	026662555
7	Tuborg	072592555

```
-- 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
```

**Print-Screen:**

The screenshot shows the Oracle SQL Developer interface. A query window is open with the following SQL command:

```
668 select * from continut_comanda;
```

The results are displayed in a table titled "Query Result". The table has three columns: ID\_PRODUS, ID\_COMANDA, and NUMAR\_PRODUSE. The data is as follows:

ID_PRODUS	ID_COMANDA	NUMAR_PRODUSE
1	1	4
2	1	2
3	2	2
4	2	1
5	3	5
6	3	4
7	4	5
8	5	4
9	5	2
10	3	3
11	2	6

-- 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
```

***Print-Screen:***

The screenshot shows the Oracle SQL Developer interface. In the top-left, there are two code snippets. The first snippet is line 632: `select * from comanda;`. The second snippet is line 633, which is empty. Below these is a toolbar with icons for Script Output, Query Result, and SQL. The SQL tab is selected. A message bar indicates "All Rows Fetched: 6 in 0.004 seconds". The main area displays a grid of data with the following columns: ID\_COMANDA, ID\_CLIENT, ID\_FACTURA, PRET, and DATA. The data is as follows:

ID_COMANDA	ID_CLIENT	ID_FACTURA	PRET	DATA
1	1	1	4	50 07-JAN-22
2	2	3	5	130 07-JAN-22
3	3	2	3	20 07-JAN-22
4	4	4	2	230 07-JAN-22
5	5	5	1	80 07-JAN-22
6	6	1	4	100 07-JAN-22

```
-- 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', 'Mirce', '0721611211'); --2
```

```
insert into client (id_client, nume, prenume, nr_telefon)
values (id_client.nextval, 'Gheorghe', 'Sebastian', '0744573419'); --3
```

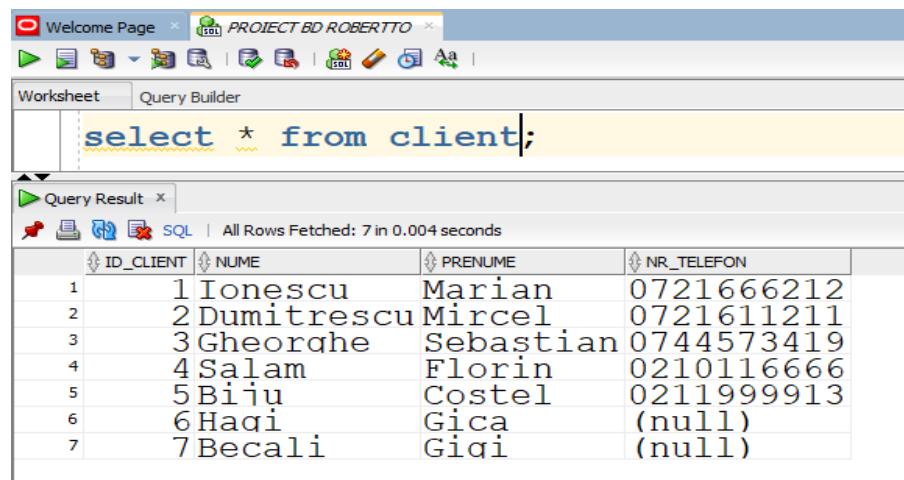
```
insert into client (id_client, nume, prenume, nr_telefon)
values (id_client.nextval, 'Salam', 'Florin', '0210116666'); --4
```

```
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:



The screenshot shows the Oracle SQL Developer interface. The top bar displays the title 'PROJECT BD ROBERTTO'. Below the title bar is a toolbar with various icons. The main area has two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab is active, containing the SQL query: 'select \* from client;'. The 'Query Result' tab is also visible below it. A message in the 'Query Result' tab says 'All Rows Fetched: 7 in 0.004 seconds'. The results are displayed in a table with columns: ID\_CLIENT, NUME, PRENUME, and NR\_TELEFON. The data is as follows:

ID_CLIENT	NUME	PRENUME	NR_TELEFON
1	Ionescu	Marian	0721666212
2	Dumitrescu	Mirce	0721611211
3	Gheorghe	Sebastian	0744573419
4	Salam	Florin	0210116666
5	Biju	Costel	0211999913
6	Hagi	Gica	(null)
7	Becali	Gigi	(null)

```
-- 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:**

The screenshot shows a MySQL Workbench interface. In the top-left pane, there are two lines of code: '520 | select \* from factura;' and '521 |'. Below this is a 'Query Result' tab with a table titled 'Query Result'. The table has four columns: ID\_FACTURA, ID\_ANGAJAT, VALOARE, and DETALII. The data is as follows:

ID_FACTURA	ID_ANGAJAT	VALOARE	DETALII
1	1	16	100 CASH
2	2	12	250 CARD
3	3	8	50 CASH
4	4	3	70 CASH
5	5	5	150 CARD

-- 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:***

The screenshot shows a SQL query window with the following content:

```
538 | select * from casier;
```

Below the query window is a "Query Result" tab showing the results of the query:

	ID_ANGAJAT	ANI_STUDIU
1	3	10
2	16	2
3	5	0
4	12	1
5	8	4

Information at the bottom of the result window: All Rows Fetched: 5 in 0.003 seconds.

```
-- 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, nume, prenume)
```

```
values (id_angajat.nextval, 1, 'Popescu', 'Robertto'); --1
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
```

```
values (id_angajat.nextval, 2, 'Escobar', 'Ricardo', to_date('15-02-21', 'dd-mm-yy')); --2
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
```

```
values (id_angajat.nextval, 4, 'Marinescu', 'Teodora', to_date('10-01-20', 'dd-mm-yy')); --3
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
```

```
values (id_angajat.nextval, 3, 'Marinescu', 'Petre', to_date('15-01-21', 'dd-mm-yy')); --4
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
values (id_angajat.nextval, 2, 'Manole', 'Alexandru', to_date('20-06-19','dd-mm-yy')); --5
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume)
values (id_angajat.nextval, 1, 'Voicu', 'Andrei'); --6
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
values (id_angajat.nextval, 3, 'Radoi', 'Raisa', to_date('15-05-20','dd-mm-yy')); --7
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
values (id_angajat.nextval, 4, 'Stan', 'Mihnea', to_date('01-01-20','dd-mm-yy')); --8
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
values (id_angajat.nextval, 5, 'Filip', 'Mihnea', to_date('21-09-20','dd-mm-yy')); --9
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
values (id_angajat.nextval, 1, 'Peste', 'Florin', to_date('25-07-20','dd-mm-yy')); --10
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
values (id_angajat.nextval, 2, 'Raducanu', 'Sorin', to_date('01-04-21','dd-mm-yy')); --11
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
values (id_angajat.nextval, 3, 'Bida', 'Marian', to_date('25-02-21','dd-mm-yy')); --12
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
values (id_angajat.nextval, 4, 'Fredo', 'Magiore', to_date('01-05-21','dd-mm-yy')); --13
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
values (id_angajat.nextval, 5, 'Cercel', 'Florin', to_date('02-09-20','dd-mm-yy')); --14
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
values (id_angajat.nextval, 4, 'Stanescu', 'Gigel', to_date('02-07-20', 'dd-mm-yy')); --15
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
values (id_angajat.nextval, 2, 'Dociu', 'Mihai', to_date('17-08-20', 'dd-mm-yy')); --16
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
values (id_angajat.nextval, 1, 'Dumitrescu', 'Florin', to_date('04-03-21', 'dd-mm-yy')); --17
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
values (id_angajat.nextval, 3, 'Sociu', 'Razvan', to_date('29-12-20', 'dd-mm-yy')); --18
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
values (id_angajat.nextval, 5, 'Scarlatescu', 'Catalin', to_date('01-03-21', 'dd-mm-yy')); --19
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume, data_angajare)
values (id_angajat.nextval, 2, 'Bontea', 'Sorin', to_date('02-03-21', 'dd-mm-yy')); --20
```

```
insert into angajat (id_angajat, id_restaurant, nume, prenume)
values (id_angajat.nextval, 1, 'Daniel', 'Andrei'); --21
```

**Print-Screen:**

The screenshot shows the Oracle SQL Developer interface. A query window titled '380 | select \* from angajat;' is open. Below it, a 'Query Result' tab is selected, showing the output of the query. The output is a table with 21 rows, each representing an employee ('angajat') with columns: ID\_ANGAJAT, ID\_RESTAURANT, NUME, PRENUME, and DATA\_ANGAJARE. The data includes various names like Popescu, Escobar, Marinescu, etc., and their employment details.

ID_ANGAJAT	ID_RESTAURANT	NUME	PRENUME	DATA_ANGAJARE
1	1	1 Popescu	Robertto	07-JAN-22
2	2	2 Escobar	Ricardo	15-FEB-21
3	3	4 Marinescu	Teodora	10-JAN-20
4	4	3 Marinescu	Petre	15-JAN-21
5	5	2 Manole	Alexandru	20-JUN-19
6	6	1 Voicu	Andrei	07-JAN-22
7	7	3 Radoi	Raisa	15-MAY-20
8	8	4 Stan	Mihnea	01-JAN-20
9	9	5 Filip	Mihnea	21-SEP-20
10	10	1 Peste	Florin	25-JUL-20
11	11	2 Raducanu	Sorin	01-APR-21
12	12	3 Bida	Marian	25-FEB-21
13	13	4 Fredo	Magiore	01-MAY-21
14	14	5 Cercel	Florin	02-SEP-20
15	15	4 Stanescu	Gigel	02-JUL-20
16	16	2 Dociu	Mihai	17-AUG-20
17	17	1 Dumitrescu	Florin	04-MAR-21
18	18	3 Sociu	Razvan	29-DEC-20
19	19	5 Scarlatescu	Catalin	01-MAR-21
20	20	2 Bontea	Sorin	02-MAR-21
21	21	1 Daniel	Andrei	07-JAN-22

-- 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:**

The screenshot shows a MySQL Workbench interface. In the top-left pane, there is a number '644'. In the main pane, a SQL query is being run: 'select \* from preparare;'. Below the query, the 'Query Result' tab is active, showing the results of the query. The results are displayed in a table with five columns: ID\_PRODUS, ID\_COMANDA, ID\_ANGAJAT, and DURATA. The table has 10 rows of data.

ID_PRODUS	ID_COMANDA	ID_ANGAJAT	DURATA
1	1	1	17
2	1	2	9
3	2	4	19
4	3	4	19
5	4	4	19
6	5	4	17
7	3	3	35
8	3	20	75
9	1	2	25
10	4	17	70
		5	16
		2	130
		18	240

-- 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:**

The screenshot shows a MySQL Workbench interface. In the top-left pane, there is a code editor with the following SQL script:

```
400 select * from bucatar;
401
402 -- PENTRU TABELUL CASIER --
```

In the bottom-right pane, there is a "Query Result" tab showing the output of the query:

ID_ANGAJAT	NR_STELE
1	9
2	17
3	18
4	19
5	20
6	21

The status bar at the bottom indicates "All Rows Fetched: 6 in 0.045 seconds".

```
-- 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:**

The screenshot shows a MySQL query editor window. The SQL query entered is "select \* from chelner;". The results are displayed in a table titled "Query Result" with 5 rows. The columns are labeled "ID\_ANGAJAT" and "ANI\_EXPERIENTA". The data is as follows:

ID_ANGAJAT	ANI_EXPERIENTA
1	4
2	7
3	11
4	14
5	15

```
-- 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
```

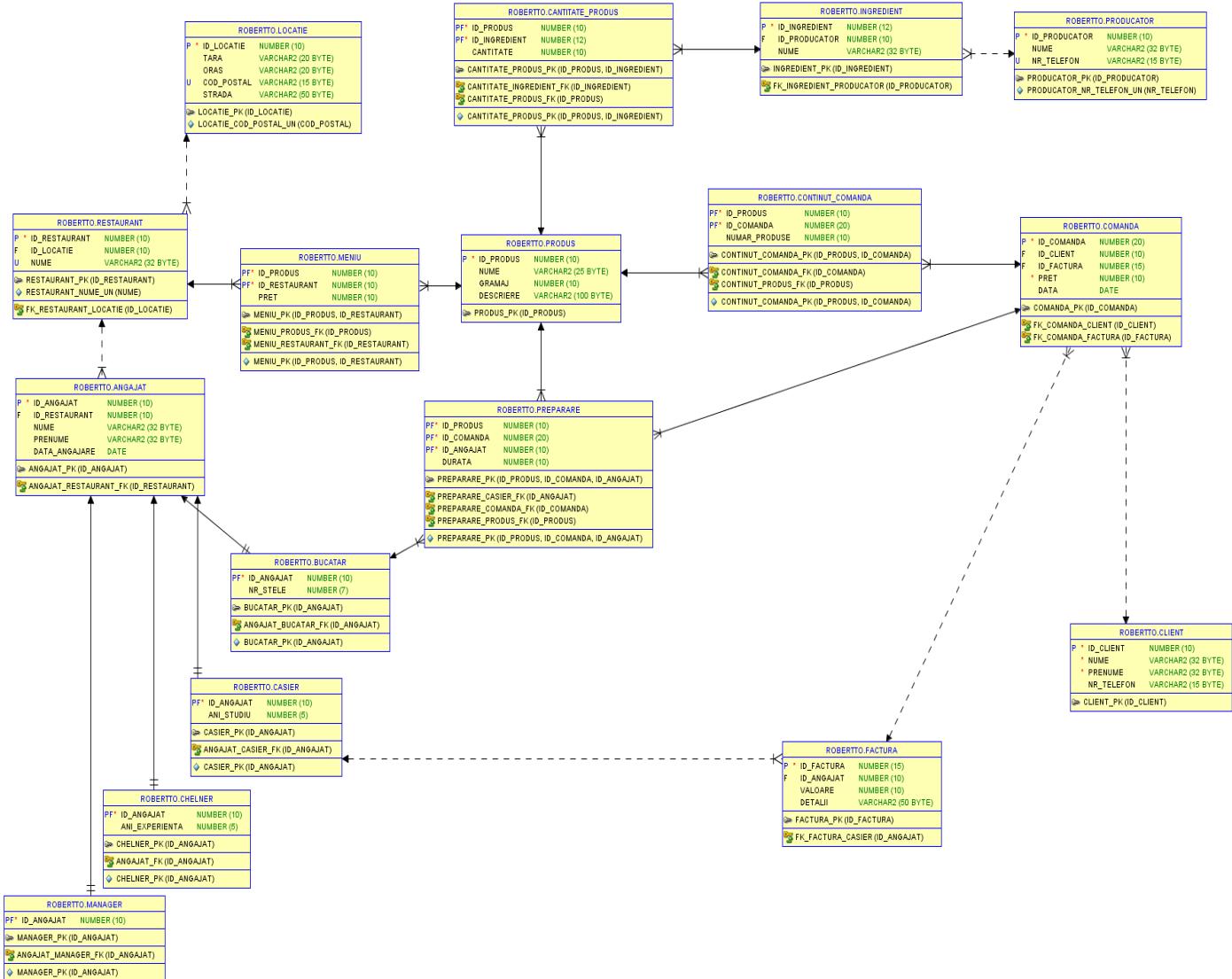
```
insert into manager (id_angajat)  
values (13); --5
```

**Print-Screen:**

The screenshot shows a MySQL Workbench interface. In the top-left corner, the number '698' is displayed. To its right is a code editor window containing the SQL query: `select * from manager;`. Below the code editor is a toolbar with icons for Refresh, Stop, Run, and SQL. The status bar indicates "All Rows Fetched: 5 in 0.003 seconds". The main area displays a table titled "Query Result" with one column labeled "ID\_ANGAJAT". The data rows are: 1, 2, 3, 4, and 5, corresponding to values 1, 2, 6, 10, and 13 respectively.

ID_ANGAJAT
1
2
3
4
5

## 5.2 Diagrama generată în sql după crearea tabelelor și inserarea datelor.



## 6. Crearea unui subprogram stocat care utilizează două tipuri de colecție studiate:

Produsele preferate pentru un client citit de la tastatură. Se afisează numele și prenumele clientului, produsele preferate 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 tabel imbricat.

```
CREATE OR REPLACE TYPE tab_imb IS
```

```
    TABLE OF NUMBER(10);
```

```
/
```

```
CREATE OR REPLACE FUNCTION afis_produse_preferate (my_id_client IN client.id_client%TYPE)
```

```
RETURN tab_imb IS
```

```
    TYPE tab_ind IS TABLE OF PLS_INTEGER INDEX BY PLS_INTEGER;
```

```
    v_contor          tab_ind;
```

```
    v_index_max       PLS_INTEGER;
```

```
    id               PLS_INTEGER;
```

```
    v_id_feluri_preferate   tab_imb;
```

```
    ok              BINARY_INTEGER := 0;
```

```
BEGIN
```

```
    v_id_feluri_preferate := tab_imb();
```

```
    FOR i IN (SELECT p.id_produs -- CURSOR IMPLICIT
```

```
        FROM produs p, continut_comanda cont, comanda cmd
```

```
        WHERE my_id_client = cmd.id_client
```

```
        AND cmd.id_comanda = cont.id_comanda
```

```
        AND cont.id_produs = p.id_produs) LOOP BEGIN
```

```
        v_contor(i.id_produs) := v_contor(i.id_produs) + 1; -- 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) := 1; -- 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

    DBMS_OUTPUT.PUT_LINE('Acest client nu a comandat nimic in viata lui');

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;

RETURN v_id_feluri_preferate;

END;
/

```

```

DECLARE
    v_id_client_citit    client.id_client%TYPE := &client_id;
    my_tab              tab_imb;
    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
    my_tab := afis_produse_preferate(v_id_client_citit);
    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;
END;

```

## *Print-Screen:*

```
702 -- Exercitiul 6
703 -- Produsele preferate pentru un client citit de la tastatura
704 -- Se afiseaza numele si prenumele clientului, produsele preferate si de cate ori le-a comandat.
705
706 CREATE OR REPLACE TYPE tab_imb IS
707     TABLE OF NUMBER(10);
708 /
709
710 CREATE OR REPLACE FUNCTION afis_produse_preferate (my_id_client IN client.id_client%TYPE)
711 RETURN tab_imb IS
712     TYPE tab_ind IS TABLE OF PLS_INTEGER INDEX BY PLS_INTEGER;
713     v_contor          tab_ind;
714     v_index_max       PLS_INTEGER;
715     id                PLS_INTEGER;
716     v_id_feluri_preferate tab_imb;
717     ok               BINARY_INTEGER := 0;
718 BEGIN
719     v_id_feluri_preferate := tab_imb();
720     FOR i IN (SELECT p.id_produs -- CURSOR IMPLICIT
721                 FROM produs p, continut_comanda cont, comanda cmd
722                 WHERE my_id_client = cmd.id_client
723                   AND cmd.id_comanda = cont.id_comanda
724                   AND cont.id_produs = p.id_produs) LOOP BEGIN
725         v_contor(i.id_produs) := v_contor(i.id_produs) + 1; -- simulez un vector de frecventa, unde tin de cate ori a fost comandat produsul
726         IF ok = 1 THEN
727             IF v_contor(i.id_produs) > v_contor(v_index_max) THEN
728                 v_index_max := i.id_produs;
729             END IF;
730         END IF;
731     END LOOP;
732     RETURN v_id_feluri_preferate;
733 END;
```

Type TAB IMB compiled

Function AFIS PRODUSE PREFERATE compiled

The screenshot shows the Oracle SQL Developer interface during the execution of a PL/SQL procedure. The code in the 'Script' tab is as follows:

```
751     id := v_contor.NEXT(id);
752   END LOOP;
753 END IF;
754 RETURN v_id_feluri_preferate;
755END;
756/
757
758DECLARE
759   v_id_client_citit      client.id%TYPE := &client_id;
760   my_tab                  tab_imb;
761   id                      PLS_INTEGER;
762   v_nume_produs          produs.name%TYPE;
763   v_name                 client.name%TYPE;
764   v_prenume              client.prenume%TYPE;
765   v_contor                PLS_INTEGER := 0;
766BEGIN
767   my_tab := afis_produse_preferate(v_id_client_citit);
768   SELECT c.name, c.prenume INTO v_name, v_prenume
769   FROM client c
770   WHERE c.id_client = v_id_client_citit;
771   DBMS_OUTPUT.PUT_LINE('Clientul ' || v_name || ' ' || v_prenume || ' are urmatoarele produse favorite:');
772   id := my_tab.FIRST; -- prima valoare din tabelul indexat (vector frecventa)
773   LOOP
774     EXIT WHEN id IS NULL;
775     SELECT p.name INTO v_nume_produs
776     FROM produs p
777     WHERE p.id_produs = my_tab(id);
778     -- afisam produsele cu numarul maxim de aparitii
779     v_contor := v_contor + 1;
780     DBMS_OUTPUT.PUT_LINE(v_contor || '.' || v_nume_produs);
781     id := my_tab.NEXT(id);
782   END LOOP;
783END;
```

The 'Script Output' tab at the bottom shows the completed task message:

```
Task completed in 2.404 seconds
```

The 'Script' tab also contains the final message:

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

In the top right corner, the 'Dms Output' tab displays the output of the DBMS\_OUTPUT.PUT\_LINE statements:

```
Clientul Gheorghe Sebastian are urmatoarele produse favorite:
1. Spaghete
2. Pizza
3. Ciocan de post
```

```

757
758  DECLARE
759    v_id_client_citit      client.id_client%TYPE := &client_id;
760    my_tab                 tab_id%TYPE;
761    id                     PLS_INTEGER;
762    v_nume_produs          produs.nume%TYPE;
763    v_nume                 client.nume%TYPE;
764    v_prenume              client.prenume%TYPE;
765    v_contor                PLS_INTEGER := 0;
766  BEGIN
767    my_tab := afis_produse_preferate(v_id_client_citit);
768    SELECT c.nume, c.prenume INTO v_nume, v_prenume
769    FROM client c
770    WHERE c.id_client = v_id_client_citit;
771    DBMS_OUTPUT.PUT_LINE('Clientul ' || v_nume || ' ' || v_prenume || ' are urmatoarele produse favorite:');
772    id := my_tab.FIRST; -- prima valoare din tabelul indexat (vector frecventa)
773    LOOP
774      EXIT WHEN id IS NULL;
775      SELECT p.nume INTO v_nume_produs
776      FROM produs p
777      WHERE p.id_produs = my_tab(id);
778      -- afisam produsele cu numarul maxim de aparitii
779      v_contor := v_contor + 1;
780      DBMS_OUTPUT.PUT_LINE(v_contor || ' ' || v_nume_produs);
781      id := my_tab.NEXT(id);
782    END LOOP;
783  END;
784
785  -- Exercitiul 7
786  -- Sa se afiseze pentru fiecare bucatar numele si prenumele acestuia, daca a preparat vreun produs de cand s-a angajat.
787  -- Daca a preparat minim un produs se va afisa date corespunzatoare legate de acesta (numele produsului, descrierea acestuia si
788  -- cat a durat sa il prepare)
789

```

Script Output: Task completed in 2.021 seconds

```

END LOOP;
END;

PL/SQL procedure successfully completed.

```

## 7. Crearea unui subprogram stocat care utilizează un tip de cursor studiat:

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).

Aici voi folosi un cursor imbricat.

```

CREATE OR REPLACE PROCEDURE afis_durata_preparare_produs
IS TYPE refcursor IS REF CURSOR;
CURSOR c IS SELECT a.id_angajat, a.nume, a.prenume, b.nr_stele,
CURSOR (SELECT prep.durata, p.nume, p.descriere, p.id_produs
FROM preparare prep, produs p
WHERE b.id_angajat = prep.id_angajat

```

```

        AND p.id_produs = prep.id_produs)

    FROM angajat a, bucatar b

    WHERE a.id_angajat = b.id_angajat;

TYPE rec IS RECORD (id_angajat.id_angajat%TYPE, dur_preparare.durata%TYPE);

TYPE tab_ind IS TABLE OF rec INDEX BY PLS_INTEGER;

t          tab_ind;

v_cursor      refcursor;

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;

v_contor       PLS_INTEGER := 0;

v_nume_bucatar_best angajat.nume%TYPE;

v_prenume_bucatar_best angajat.prenume%TYPE;

v_nr_stele_bucatar_best bucatar.nr_stele%TYPE;

BEGIN

OPEN c;

LOOP

    FETCH c INTO v_id_bucatar, v_nume_bucatar, v_prenume_bucatar, v_nr_stele_bucatar, v_cursor;

    EXIT WHEN c%NOTFOUND;

    FETCH v_cursor INTO v_durata_preparare, v_nume_produs, v_descriere_produs, v_id_produs;

    IF v_cursor%NOTFOUND THEN

        DBMS_OUTPUT.PUT_LINE('ATENTIE! Bucatarul ' || v_nume_bucatar || ' ' || v_prenume_bucatar
|| ' nu a preparat niciodata un produs!');

        DBMS_OUTPUT.NEW_LINE();
    END IF;
END LOOP;
END;

```

```

    ELSE
        DBMS_OUTPUT.PUT_LINE('Bucatarul ' || v_nume_bucatar || ' ' || v_prenume_bucatar || ' a
preparat urmatoarele produse:');
    LOOP
        EXIT WHEN v_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(v_cursor%ROWCOUNT || '. produsul: ' || v_nume_produs || ' cu
descrierea ' || 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 v_cursor INTO v_durata_preparare, v_nume_produs, v_descriere_produs, v_id_produs;
    END LOOP;
    END IF;
END LOOP;
CLOSE c;
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.');

```

```

END LOOP;

END;

/

BEGIN

    afis_durata_preparare_produs();

END;

/

```

**Print-Screen:**

The screenshot shows the Oracle SQL Developer interface with the 'Script Output' tab selected. The script area contains the code for creating a procedure named 'afis\_durata\_preparare\_produs'. The code uses two nested cursors to select data from three tables: 'angajat', 'bucatar', and 'produs'. It defines record types for each cursor and a table type for the result set. The procedure begins with an 'OPEN c;' statement. The output window at the bottom shows the message 'PL/SQL procedure successfully completed.' and indicates the task was completed in 0.073 seconds.

```

785 -- Exercitiul 7
786 -- Sa se afiseze pentru fiecare bucatar numele si prenumele acestuia, daca a preparat vreun produs de cand s-a angajat.
787 -- Daca a preparat minim un produs se va afisa date corespunzatoare legate de acesta (numele produsului, descrierea acestuia si
788 -- cat a durat sa il prepare)
789
790 CREATE OR REPLACE PROCEDURE afis_durata_preparare_produs
791 IS TYPE refcursor IS REF CURSOR;
792     CURSOR c IS SELECT a.id_angajat, a.nume, a.prenume, b.nr_stele,
793                     CURSOR (SELECT prep.durata, p.nume, p.descriere, p.id_produs
794                         FROM preparare prep, produs p
795                         WHERE b.id_angajat = prep.id_angajat
796                             AND p.id_produs = prep.id_produs)
797                     FROM angajat a, bucatar b
798                     WHERE a.id_angajat = b.id_angajat;
799     TYPE rec IS RECORD (id_angajat.id_angajat%TYPE, dur preparare.durata%TYPE);
800     TYPE tab_ind IS TABLE OF rec INDEX BY PLS_INTEGER;
801     t                      tab_ind;
802     v_cursor                refcursor;
803     v_id_bucatar            angajat.id_angajat%TYPE;
804     v_nume_bucatar          angajat.nume%TYPE;
805     v_prenume_bucatar       angajat.prenume%TYPE;
806     v_nr_stele_bucatar     bucatar.nr_stele%TYPE;
807     v_durata_preparare     preparare.durata%TYPE;
808     v_id_produs             produs.id_produs%TYPE;
809     v_nume_produs           produs.nume%TYPE;
810     v_descriere_produs     produs.descriere%TYPE;
811     v_contor                PLS_INTEGER := 0;
812     v_nume_bucatar_best    angajat.nume%TYPE;
813     v_prenume_bucatar_best angajat.prenume%TYPE;
814     v_nr_stele_bucatar_best bucatar.nr_stele%TYPE;
815
816 BEGIN
     OPEN c;

```

Script Output | Task completed in 0.073 seconds  
PL/SQL procedure successfully completed.

Procedure AFIS\_DURATA\_PREPARARE\_PRODUS compiled

```

826    LOOP
827        EXIT WHEN v_cursor%NOTFOUND;
828        DBMS_OUTPUT.PUT_LINE(v_cursor%ROWCOUNT || '. produsul: ' || v_nume_produs || ' cu descrierea ' || v_descriere);
829        IF t.EXISTS (v_id_produs) THEN
830            IF v_durata_preparare < t(v_id_produs).dur THEN
831                t(v_id_produs) := rec(v_id_bucatar, v_durata_preparare);
832            END IF;
833        ELSE
834            t(v_id_produs) := rec(v_id_bucatar, v_durata_preparare);
835        END IF;
836        FETCH v_cursor INTO v_durata_preparare, v_nume_produs, v_descriere_produs, v_id_produs;
837        END LOOP;
838    END LOOP;
839    CLOSE c;
840    FOR i IN t.FIRST..t.LAST LOOP
841        SELECT nume INTO v_nume_produs
842        FROM produs
843        WHERE id_produs = i;
844
845        SELECT a.nume, a.prenume INTO v_nume_bucatar, v_prenume_bucatar
846        FROM angajat a, bucatar b
847        WHERE a.id_angajat = t(i).id_ang
848        AND a.id_angajat = b.id_angajat;
849        DBMS_OUTPUT.PUT_LINE('Bucatarul ' || v_nume_bucatar || ' ' || v_prenume_bucatar || ' a preparat produsul ' || v_nume_produs);
850    END LOOP;
851    END;
852    /
853
854
855    BEGIN
856        afis_durata_preparare_produs();
857    END;
858
```

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

## 8. Crearea unui subprogram stocat de tip funcție, care utilizează într-o singură comandă SQL minim 3 tabele (tratarea și punerea în evidență a exceptiilor):

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. 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;
v_exista                     PLS_INTEGER := 0;
t                            tab_ind;

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;
    END;
  END LOOP;
END;

```

```

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
    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;

RETURN t(my_id_client);

EXCEPTION
WHEN invalid THEN
    RAISE_APPLICATION_ERROR(-20021, 'Nu a fost gasit niciun client cu id-ul introdus!');
WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20022, 'Alta eroare!');

END;

```

/

## DECLARE

```
v_id_client_citit    client.id_client%TYPE := &client_id;  
v_resultat          PLS_INTEGER;
```

BEGIN

```
v_resultat := profit_client(v_id_client_citit);
```

IF v\_resultat > 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;

1

## ***Print-Screen:***

```

860 -- Exercitiul 8
861 -- Sa se afiseze profitul generat de un client citit de la tastatura, de-a lungul timpului,
862 -- Daca acesta a beneficiat de reduceri de-a lungul timpului se va afisa valoarea totala a acestor reduceri.
863 -- Daca profitul adus de-a lungul timpului este unul negativ inseamna ca clientul a beneficiat de prea multe reduceri
864 -- si se va afisa un mesaj corespunzator.
865 -- De asemenea se va afisa profitul maxim oferit de un client si profitul obtinut de companie pana la momentul actual.
866
867 CREATE OR REPLACE FUNCTION profit_client (my_id_client IN client.id_client%TYPE)
868 RETURN PLS_INTEGER IS
869   TYPE tab_ind IS TABLE OF PLS_INTEGER INDEX BY PLS_INTEGER;
870   v_nume_client           client.nume%TYPE;
871   v_prenume_client         client.nume%TYPE;
872   v_nr_telefon_client     client.nr_telefon%TYPE;
873   v_profit_maxim_companie PLS_INTEGER := 0;
874   v_profit_maxim_client   PLS_INTEGER := 0;
875   v_benef_red              BINARY_INTEGER := 0;
876   v_valoare_red            PLS_INTEGER := 0;
877   invalid                  EXCEPTION;
878   v_exista                PLS_INTEGER := 0;
879   t                         tab_ind;
880
881 BEGIN
882   SELECT COUNT(*) INTO v_exista
883   FROM client
884   WHERE id_client = my_id_client;
885   IF v_exista = 0 THEN
886     RAISE invalid;
887   END IF;
888
```

Function PROFIT CLIENT compiled

PROJECT BD ROBERTTO

Worksheet Query Builder

```

921     DBMS_OUTPUT.PUT_LINE('Nu beneficiaza de reduceri!');
922   END IF;
923   RETURN t(my_id_client);
924 EXCEPTION
925   WHEN invalid THEN
926     RAISE_APPLICATION_ERROR(-20021, 'Nu a fost gasit niciun client cu id-ul introdus!');
927   WHEN OTHERS THEN
928     RAISE_APPLICATION_ERROR(-20022, 'Alta eroare!');
929 END;
930 /
931
932 DECLARE
933   v_id_client_citit    client.id_client%TYPE := &client_id;
934   v_rezultat            PLS_INTEGER;
935
936 BEGIN
937   v_rezultat := profit_client(v_id_client_citit);
938   IF v_rezultat > 0 THEN
939     DBMS_OUTPUT.PUT_LINE('El a contribuit la profitul companiei pana acum cu ' || v_rezultat || ' lei.');
940   ELSE
941     DBMS_OUTPUT.PUT_LINE('Acesta a beneficiat de prea multe reduceri! Aducand un profit negativ companiei: ' || v_
942   END IF;
943 END;
944 /
945
946 -- Exercitiul 9
947 -- Sa se afiseze ce cantitate ingrediente contine fiecare produs, din fiecare comanda,
948 -- furnizate de un producator citit de la tastatura.
949
950 CREATE OR REPLACE PROCEDURE mai_multe_ingrediente_producator(my_id_comanda IN comanda.id_comanda%TYPE, my_id_produs IN
951 IS
952   TYPE rec IS RECORD (v_cantitate_produs cantitate_produs.cantitate%TYPE, v_nume_ingredient ingredient.nume%TYPE);
953   TYPE tab_ind IS TABLE OF rec INDEX BY PLS_INTEGER;
954
955   v_comanda comanda%ROWTYPE;
956   v_produs producator%ROWTYPE;
957   v_rec record;
958   v_tab tab_ind;
959
960   CURSOR cur_comanda IS SELECT * FROM comanda WHERE id_comanda = my_id_comanda;
961   CURSOR cur_produs IS SELECT * FROM producator WHERE id_producator = my_id_produs;
962
963   FOR cur_comanda LOOP
964     v_comanda := cur_comanda%ROWTYPE;
965     FOR cur_produs LOOP
966       v_produs := cur_produs%ROWTYPE;
967       v_rec.v_cantitate_produs := 0;
968       FOR i IN 1..v_comanda.nr_ingredienti LOOP
969         IF v_comanda.ingredient(i).id_ingredient = v_produs.id_ingredient THEN
970           v_rec.v_cantitate_produs := v_rec.v_cantitate_produs + v_comanda.ingredient(i).cantitate;
971         END IF;
972       END LOOP;
973       v_rec.v_nume_ingredient := v_produs.nume;
974       v_tab.extend;
975       v_tab(v_tab.last) := v_rec;
976     END LOOP;
977   END LOOP;
978
979   FOR i IN 1..v_tab.COUNT LOOP
980     DBMS_OUTPUT.PUT_LINE(v_tab(i).v_nume_ingredient || ': ' || v_tab(i).v_cantitate_produs);
981   END LOOP;
982
983 END;
984
985 PL/SQL procedure successfully completed.

```

Script Output X | Task completed in 2.402 seconds

Pentru id\_client = 100.

```

932 DECLARE
933   v_id_client_citit    client.id_client%TYPE := &client_id;
934   v_rezultat            PLS_INTEGER;
935
936 BEGIN
937   v_rezultat := profit_client(v_id_client_citit);
938   IF v_rezultat > 0 THEN
939     DBMS_OUTPUT.PUT_LINE('El a contribuit la profitul companiei pana acum cu ' || v_rezultat || ' lei.');
940   ELSE
941     DBMS_OUTPUT.PUT_LINE('Acesta a beneficiat de prea multe reduceri! Aducand un profit negativ companiei: ' || v_
942   END IF;
943 END;
944 /
945
946 -- Exercitiul 9
947 -- Sa se afiseze ce cantitate ingrediente contine fiecare produs, din fiecare comanda,
948 -- furnizate de un producator citit de la tastatura.
949
950 CREATE OR REPLACE PROCEDURE mai_multe_ingrediente_producator(my_id_comanda IN comanda.id_comanda%TYPE, my_id_produs IN
951 IS
952   TYPE rec IS RECORD (v_cantitate_produs cantitate_produs.cantitate%TYPE, v_nume_ingredient ingredient.nume%TYPE);
953   TYPE tab_ind IS TABLE OF rec INDEX BY PLS_INTEGER;
954
955   v_comanda comanda%ROWTYPE;
956   v_produs producator%ROWTYPE;
957   v_rec record;
958   v_tab tab_ind;
959
960   CURSOR cur_comanda IS SELECT * FROM comanda WHERE id_comanda = my_id_comanda;
961   CURSOR cur_produs IS SELECT * FROM producator WHERE id_producator = my_id_produs;
962
963   FOR cur_comanda LOOP
964     v_comanda := cur_comanda%ROWTYPE;
965     FOR cur_produs LOOP
966       v_produs := cur_produs%ROWTYPE;
967       v_rec.v_cantitate_produs := 0;
968       FOR i IN 1..v_comanda.nr_ingredienti LOOP
969         IF v_comanda.ingredient(i).id_ingredient = v_produs.id_ingredient THEN
970           v_rec.v_cantitate_produs := v_rec.v_cantitate_produs + v_comanda.ingredient(i).cantitate;
971         END IF;
972       END LOOP;
973       v_rec.v_nume_ingredient := v_produs.nume;
974       v_tab.extend;
975       v_tab(v_tab.last) := v_rec;
976     END LOOP;
977   END LOOP;
978
979   FOR i IN 1..v_tab.COUNT LOOP
980     DBMS_OUTPUT.PUT_LINE(v_tab(i).v_nume_ingredient || ': ' || v_tab(i).v_cantitate_produs);
981   END LOOP;
982
983 END;
984
985 Error report -
ORA-20021: Nu a fost gasit niciun client cu id-ul introdus!
ORA-06512: at "ROBERTTO.PROFIT_CLIENT", line 60
ORA-06512: at line 6

```

Script Output X | Task completed in 2.574 seconds

## **9. Crearea unui subprogram stocat de tip funcție, care utilizează într-o singură comandă SQL minim 5 tabele (tratarea și punerea în evidență a exceptiilor, NO\_DATA\_FOUND și TOO\_MANY\_ROWS):**

Să se afișeze ce cantitate de ingrediente conține fiecare produs, din fiecare comandă, furnizată de un producător citit de la tastatură.

```
CREATE OR REPLACE PROCEDURE mai_multe_ingredientele_producator(my_id_comanda IN  
comanda.id_comanda%TYPE, my_id_produs IN produs.id_produs%TYPE, my_id_producator IN  
producator.id_producator%TYPE, my_numar_produse_comanda  
continut_comanda.numar_produse%TYPE)
```

IS

```
    TYPE rec IS RECORD (v_cantitate_produs cantitate_produs.cantitate%TYPE, v_nume_ingredient  
ingredient.nume%TYPE);
```

```
    TYPE tab_ind IS TABLE OF rec INDEX BY PLS_INTEGER;
```

```
    t tab_ind;
```

```
BEGIN
```

```
    SELECT cnt.cantitate, i.nume BULK COLLECT INTO t  
    FROM cantitate_produs cnt, ingredient i, producator prod, produs p, continut_comanda cmd
```

```
    WHERE cmd.id_comanda = my_id_comanda
```

```
    AND cmd.id_produs = my_id_produs
```

```
    AND p.id_produs = cmd.id_produs
```

```
    AND cnt.id_produs = p.id_produs
```

```
    AND i.id_ingredient = cnt.id_ingredient
```

```
    AND prod.id_producator = i.id_producator
```

```
    AND my_id_producator = prod.id_producator;
```

```
    FOR i IN 1..t.LAST LOOP
```

```
        DBMS_OUTPUT.PUT_LINE('-> ' || my_numar_produse_comanda*t(i).v_cantitate_produs || 'g de '  
        ||t(i).v_nume_ingredient || '.');
```

```
    END LOOP;
```

```
END;
```

```
/
```

```

CREATE OR REPLACE PROCEDURE afis_cantitate_ingredient_producator_produs(my_id_producator IN
producator.id_producator%TYPE) IS

CURSOR mycursor IS (SELECT p.id_produs, p.nume, cmd.numar_produse, cmd.id_comanda
FROM produs p, continut_comanda cmd
WHERE p.id_produs=cmd.id_produs);

TYPE tab_ind IS TABLE OF ingredient.nume%TYPE INDEX BY PLS_INTEGER;

v_id_produs          produs.id_produs%TYPE;
v_id_comanda         continut_comanda.id_comanda%TYPE;
v_nume_produs        produs.nume%TYPE;
v_numar_produse      continut_comanda.numar_produse%TYPE;
v_cantitate_produs   cantitate_produs.cantitate%TYPE;
v_nume_ingredient    ingredient.nume%TYPE;
v_nume_producator   producator.nume%TYPE;
v_nr_telefon_producator   producator.nr_telefon%TYPE;
lista_ingrediente_producator   tab_ind;

BEGIN

SELECT nume, nr_telefon INTO v_nume_producator, v_nr_telefon_producator
FROM producator
WHERE id_producator = my_id_producator;

SELECT nume BULK COLLECT INTO lista_ingrediente_producator
FROM ingredient
WHERE id_producator = my_id_producator;

DBMS_OUTPUT.PUT_LINE('Numele producatorului selectat este ' || v_nume_producator || '.');
DBMS_OUTPUT.PUT_LINE('El poate fi contactat la numarul de telefon: ' || v_nr_telefon_producator || '.');
DBMS_OUTPUT.PUT_LINE('Acesta produce urmatoarele ingrediente:');
FOR i IN 1..lista_ingrediente_producator.LAST LOOP
DBMS_OUTPUT.PUT_LINE('-> ' || lista_ingrediente_producator(i) || ';');
END LOOP;

```

```

END LOOP;

DBMS_OUTPUT.NEW_LINE();

OPEN mycursor;

LOOP

  FETCH mycursor INTO v_id_produs,v_nume_produs, v_numar_produse, v_id_comanda;
  EXIT WHEN mycursor%NOTFOUND;

  DBMS_OUTPUT.PUT_LINE('-- ' || v_nume_produs || ' are de la producatorul ' ||
v_nume_producator || ' in comanda cu id-ul ' || v_id_comanda || ' urmatorul ingredient:');
  BEGIN

    SELECT cnt.cantitate, i.nume INTO v_cantitate_produs, v_nume_ingredient
    FROM cantitate_produs cnt, ingredient i, producator prod, produs p, continut_comanda cmd
    WHERE cmd.id_comanda = v_id_comanda
    AND cmd.id_produs = v_id_produs
    AND p.id_produs = cmd.id_produs
    AND cnt.id_produs = p.id_produs
    AND i.id_ingredient = cnt.id_ingredient
    AND prod.id_producator = i.id_producator
    AND my_id_producator = prod.id_producator;

    DBMS_OUTPUT.PUT_LINE('-> ' || v_numar_produse*v_cantitate_produs || 'g de ' ||
v_nume_ingredient || '.');

    DBMS_OUTPUT.NEW_LINE();

  EXCEPTION

    WHEN NO_DATA_FOUND THEN

      DBMS_OUTPUT.PUT_LINE('ATENTIE! Nu exista niciun ingredient al acestui producator in acest
preparat!');

      DBMS_OUTPUT.NEW_LINE();

    WHEN TOO_MANY_ROWS THEN

      DBMS_OUTPUT.PUT_LINE('ATENTIE! Mai multe ingrediente ale acestui producator detectate in
acest preparat!');

    BEGIN

```

```

        mai_multe_ingrediente_producator(v_id_comanda, v_id_produs, my_id_producator,
v_numar_produse);

    END;

    DBMS_OUTPUT.NEW_LINE();

    END;

    END LOOP;

    CLOSE mycursor;

END;

/

```

DECLARE

```
v_id_producator producator.id_producator%TYPE := &producator_id;
```

BEGIN

```
afis_cantitate_ingredient_producator_produs(v_id_producator);
```

END;

/

### ***Print-Screen:***

The screenshot shows the Oracle SQL Developer interface. The top menu bar includes 'File', 'Edit', 'Tools', 'Help', and a status bar indicating '0.117 seconds'. The main area is a 'Worksheet' tab with 'Query Builder' selected. The code is displayed in the query builder window.

```

947 -- Sa se afiseze ce cantitate ingrediente contine fiecare produs, din fiecare comanda,
948 -- furnizat de un producator citit de la tastatura.
949
950 CREATE OR REPLACE PROCEDURE mai_multe_ingrediente_producator(my_id_comanda IN comanda.id_comanda%TYPE, my_id_produs
951 IS
952     TYPE rec IS RECORD (v_cantitate_produs cantitate_produs.cantitate%TYPE, v_nume_ingredient ingredient.nume%TYPE);
953     TYPE tab_ind IS TABLE OF rec INDEX BY PLS_INTEGER;
954     t tab_ind;
955
956 BEGIN
957     SELECT cnt.cantitate, i.nume BULK COLLECT INTO t
958     FROM cantitate_produs cnt, ingredient i, producator prod, produs p, continut_comanda cmd
959     WHERE cmd.id_comanda = my_id_comanda
960     AND cmd.id_produs = my_id_produs
961     AND p.id_produs = cmd.id_produs
962     AND cnt.id_produs = p.id_produs
963     AND i.id_ingredient = cnt.id_ingredient
964     AND prod.id_producator = i.id_producator
965     AND my_id_producator = prod.id_producator;
966     FOR i IN 1..t.LAST LOOP
967         DBMS_OUTPUT.PUT_LINE(' -> ' || my_numar_produse_comanda*t(i).v_cantitate_produs || 'g de ' || t(i).v_nume_ingredient);
968     END LOOP;
969
970
971 CREATE OR REPLACE PROCEDURE afis_cantitate_ingredient_producator_produs(my_id_producator IN producator.id_producator%
972     CURSOR mycursor IS (SELECT p.id_produs, p.nume, cmd.numar_produse, cmd.id_comanda
973     FROM produs p, continut_comanda cmd
974     WHERE p.id_produs=cmd.id_produs);
975     TYPE tab_ind IS TABLE OF ingredient.nume%TYPE INDEX BY PLS_INTEGER;
976     v_id_produs
977     v_id_comanda

```

The bottom section shows the 'Script Output' tab with the message 'Task completed in 0.117 seconds'.

```

Procedure MAI_MULTE_INGREDIENTE_PRODUCATOR compiled
Procedure AFIS_CANTITATE_INGREDIENT_PRODUCATOR_PRODUS compiled

```

PROJECT BD ROBERTO

Worksheet Query Builder

```

1025      DBMS_OUTPUT.NEW_LINE();
1026      END;
1027      END LOOP;
1028      CLOSE mycursor;
1029  END;
1030 /
1031
1032DECLARE
1033      v_id_producator_producator.id_producator%TYPE := &producator_id;
1034  BEGIN
1035      afis_cantitate_ingredient_producator_produs(v_id_producator);
1036  END;
1037 /
```

-- Exercitiul 10

```

1040 CREATE OR REPLACE TRIGGER trig_vacanta BEFORE
1041      INSERT ON comanda
1042 DECLARE
1043      ziua NUMBER := extract(DAY FROM sysdate);
1044      luna NUMBER := extract(MONTH FROM sysdate);
1045  BEGIN
1046      IF ( ziua = 25 AND luna = 12 ) OR ( ziua = 1 AND luna = 1 ) THEN
1047          raise_application_error(-20067, 'Astazi nu se pot plasa comenzi, este zi libera!');
1048      END IF;
1049  END;
1050 /
1051
1052 insert into comanda (id_comanda, id_client, id_factura, pret)
1053 values (id_comanda.nextval, 1, 4, 50); --1
1054
1055 -- Exercitiul 11
```

PL/SQL procedure successfully completed.

Script Output

```

V_id_producator_producator.id_producator%TYPE := 2;
BEGIN
    afis_cantitate_ingredient_producator_produs(v_id_producator);
END;
```

PL/SQL procedure successfully completed.

Doms Output

PROJECT BD ROBERTO

Numele producatorului selectat este Olivers.  
El poate fi contactat la numarul de telefon: 0754754318.  
Acesta produce urmatoarele ingrediente:  
> bors;  
> parmezan;  
> cascaval;

-- Spaghete are de la producatorul Olivers in comanda cu id-ul 4 urmatorul ingredient:  
ATENTIE! Nu exista niciun ingredient al acestui producator in acest preparat!

-- Spaghete are de la producatorul Olivers in comanda cu id-ul 2 urmatorul ingredient:  
ATENTIE! Nu exista niciun ingredient al acestui producator in acest preparat!

-- Pizza are de la producatorul Olivers in comanda cu id-ul 2 urmatorul ingredient:  
ATENTIE! Mai multe ingrediente ale acestui producator detectate in acest preparat!  
> 50g de parmezan.  
> 35g de cascaval.

-- Pizza are de la producatorul Olivers in comanda cu id-ul 1 urmatorul ingredient:  
ATENTIE! Mai multe ingrediente ale acestui producator detectate in acest preparat!  
> 50g de parmezan.  
> 35g de cascaval.

-- Paste bolognese are de la producatorul Olivers in comanda cu id-ul 5 urmatorul ingredient:  
> 55g de bors.

-- Paste bolognese are de la producatorul Olivers in comanda cu id-ul 4 urmatorul ingredient:  
> 55g de bors.

-- Sarmale are de la producatorul Olivers in comanda cu id-ul 5 urmatorul ingredient:  
ATENTIE! Nu exista niciun ingredient al acestui producator in acest preparat!

-- Ciorba de post are de la producatorul Olivers in comanda cu id-ul 4 urmatorul ingredient:  
> 350g de bors.

-- Ciorba de post are de la producatorul Olivers in comanda cu id-ul 2 urmatorul ingredient:  
> 350g de bors.

-- Paste bolognese are de la producatorul Olivers in comanda cu id-ul 3 urmatorul ingredient:  
> 55g de bors.

-- Pizza are de la producatorul Olivers in comanda cu id-ul 6 urmatorul ingredient:  
ATENTIE! Mai multe ingrediente ale acestui producator detectate in acest preparat!  
> 100g de parmezan.  
> 70g de cascaval.

Compiler - Log

1025 DBMS\_OUTPUT.NEW\_LINE();
1026 END;
1027 END LOOP;
1028 CLOSE mycursor;
1029 END;
1030 /
1031
1032~~DECLARE~~
1033 v\_id\_producator\_producator.id\_producator%TYPE := &producator\_id;
1034 BEGIN
1035 afis\_cantitate\_ingredient\_producator\_produs(v\_id\_producator);
1036 END;
1037 /

-- Exercitiul 10

```

1040 CREATE OR REPLACE TRIGGER trig_vacanta BEFORE
1041      INSERT ON comanda
1042 DECLARE
1043      ziua NUMBER := extract(DAY FROM sysdate);
1044      luna NUMBER := extract(MONTH FROM sysdate);
1045  BEGIN
1046      IF ( ziua = 25 AND luna = 12 ) OR ( ziua = 1 AND luna = 1 ) THEN
1047          raise_application_error(-20067, 'Astazi nu se pot plasa comenzi, este zi libera!');
1048      END IF;
1049  END;
1050 /
1051
1052 insert into comanda (id_comanda, id_client, id_factura, pret)
1053 values (id_comanda.nextval, 1, 4, 50); --1
1054
1055 -- Exercitiul 11
```

PL/SQL procedure successfully completed.

Script Output

```

V_id_producator_producator.id_producator%TYPE := 3;
BEGIN
    afis_cantitate_ingredient_producator_produs(v_id_producator);
END;
```

PL/SQL procedure successfully completed.

Doms Output

PROJECT BD ROBERTO

Numele producatorului selectat este FUCHS.  
El poate fi contactat la numarul de telefon: 021592555.  
Acesta produce urmatoarele ingrediente:  
> condimente;

-- Spaghete are de la producatorul FUCHS in comanda cu id-ul 4 urmatorul ingredient:  
> 100g de condimente.

-- Spaghete are de la producatorul FUCHS in comanda cu id-ul 2 urmatorul ingredient:  
> 50g de condimente.

-- Pizza are de la producatorul FUCHS in comanda cu id-ul 2 urmatorul ingredient:  
> 60g de condimente.

-- Pizza are de la producatorul FUCHS in comanda cu id-ul 1 urmatorul ingredient:  
> 60g de condimente.

-- Paste bolognese are de la producatorul FUCHS in comanda cu id-ul 5 urmatorul ingredient:  
> 200g de condimente.

-- Paste bolognese are de la producatorul FUCHS in comanda cu id-ul 4 urmatorul ingredient:  
> 200g de condimente.

-- Sarmale are de la producatorul FUCHS in comanda cu id-ul 5 urmatorul ingredient:  
> 100g de condimente.

-- Ciorba de post are de la producatorul FUCHS in comanda cu id-ul 4 urmatorul ingredient:  
ATENTIE! Nu exista niciun ingredient al acestui producator in acest preparat!

-- Ciorba de post are de la producatorul FUCHS in comanda cu id-ul 2 urmatorul ingredient:  
ATENTIE! Nu exista niciun ingredient al acestui producator in acest preparat!

-- Paste bolognese are de la producatorul FUCHS in comanda cu id-ul 3 urmatorul ingredient:  
> 200g de condimente.

-- Pizza are de la producatorul FUCHS in comanda cu id-ul 6 urmatorul ingredient:  
> 120g de condimente.

Compiler - Log

## 10. 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 eroare 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:

The screenshot shows the Oracle SQL Developer interface with the SQL worksheet tab active. The code area contains the PL/SQL code for creating the trigger. The code is as follows:

```
1038 -- Exercitiul 10
1039 -- Sa se creeze un Trigger care nu permite clientilor sa plaseze comenzi de Craciun, 25 decembrie si
1040 -- astfel incat inainte de a insera in tabelul comanda sa se arunce o eroare daca ziua este una
1041 -- din cele mentionate anterior.
1042
1043 CREATE OR REPLACE TRIGGER trig_vacanta BEFORE
1044     INSERT ON comanda
1045 DECLARE
1046     ziua    NUMBER := extract(DAY FROM sysdate);
1047     luna    NUMBER := extract(MONTH FROM sysdate);
1048 BEGIN
1049     IF ( ziua = 25 AND luna = 12 ) OR ( ziua = 1 AND luna = 1 ) THEN
1050         raise_application_error(-20067, 'Astazi nu se pot plasa comenzi, este zi libera!');
1051     END IF;
1052 END;
1053 /
1054
1055 insert into comanda (id_comanda, id_client, id_factura, pret)
```

Below the code, the "Script Output" window shows the results of the execution:

```
Script Output x
Task completed in 0.059 seconds
PL/SQL procedure successfully completed.

Trigger TRIG_VACANTA compiled
```

## 10.1 Declanșarea acestuia:

```
insert into comanda (id_comanda, id_client, id_factura, pret)  
values (id_comanda.nextval, 1, 4, 50);
```

### Print-Screen:

The screenshot shows the Oracle SQL Developer interface with a code editor and a script output window.

```
1039 -- Exercitiul 10  
1040 -- Sa se creeze un Trigger care nu permite clientilor sa plaseze comenzi de Craciun, 25 decembrie si de Anul Nou, 1 ianuarie;  
1041 -- astfel incat inainte de a insera in tabelul comanda sa se arunce o eroarea daca ziua este una din cele mentionate anterior.  
1042  
1043 CREATE OR REPLACE TRIGGER trig_vacanta BEFORE  
1044     INSERT ON comanda  
1045 DECLARE  
1046     ziua    NUMBER := extract(DAY FROM sysdate);  
1047     luna    NUMBER := extract(MONTH FROM sysdate);  
1048 BEGIN  
1049     IF ( ziua = 25 AND luna = 12 ) OR ( ziua = 1 AND luna = 1 ) THEN  
1050         raise_application_error(-20067, 'Astazi nu se pot plasa comenzi, este zi libera!');  
1051     END IF;  
1052 END;  
1053 /  
1054  
1055 insert into comanda (id_comanda, id_client, id_factura, pret)  
1056 values (id_comanda.nextval, 1, 4, 50);  
1057  
1058 -- Exercitiul 11
```

Script Output x | Task completed in 0.043 seconds

Trigger TRIG\_VACANTA compiled

1 row inserted.

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

The screenshot shows the Oracle SQL Developer interface with a code editor and a script output window.

```
1039 -- Exercitiul 10  
1040 -- Sa se creeze un Trigger care nu permite clientilor sa plaseze comenzi de Craciun, 25 decembrie si de Anul Nou, 1 ianuarie;  
1041 -- astfel incat inainte de a insera in tabelul comanda sa se arunce o eroarea daca ziua este una din cele mentionate anterior.  
1042  
1043 CREATE OR REPLACE TRIGGER trig_vacanta BEFORE  
1044     INSERT ON comanda  
1045 DECLARE  
1046     ziua    NUMBER := extract(DAY FROM sysdate);  
1047     luna    NUMBER := extract(MONTH FROM sysdate);  
1048 BEGIN  
1049     IF ( ziua = 25 AND luna = 12 ) OR ( ziua = 7 AND luna = 1 ) THEN  
1050         raise_application_error(-20067, 'Astazi nu se pot plasa comenzi, este zi libera!');  
1051     END IF;  
1052 END;  
1053 /  
1054  
1055 insert into comanda (id_comanda, id_client, id_factura, pret)  
1056 values (id_comanda.nextval, 1, 4, 50);  
1057  
1058 -- Exercitiul 11  
1059 CREATE OR REPLACE TRIGGER trig_produs_in_menu FOR  
1060     INSERT OR UPDATE OF id_produs ON continut_comanda  
1061 COMPOUND TRIGGER
```

Script Output x | Query Result x | Task completed in 0.046 seconds

```
insert into comanda (id_comanda, id_client, id_factura, pret)  
values (id_comanda.nextval, 1, 4, 50)  
Error report -  
ORA-20067: Astazi nu se pot plasa comenzi, este zi libera!  
ORA-06512: at "ROBERTTO.TRIG_VACANTA", line 6  
ORA-04088: error during execution of trigger 'ROBERTTO.TRIG_VACANTA'
```

## 11. 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_menu 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('Aces 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_menui;
/

```

**Print-Screen:**

```

1058 -- Exercitiul 11
1059 CREATE OR REPLACE TRIGGER trig_produs_in_menui FOR
1060   INSERT OR UPDATE OF id_produs ON continut_comanda
1061 COMPOUND TRIGGER
1062   v_id_restaurant meniu.id_restaurant%TYPE;
1063   BEFORE STATEMENT IS BEGIN
1064     dbms_output.put_line('Verificare daca produs exista in meniu');
1065   END BEFORE STATEMENT;
1066   BEFORE EACH ROW IS BEGIN
1067     SELECT
1068       id_restaurant
1069     INTO v_id_restaurant
1070     FROM
1071       meniu m
1072     WHERE
1073       m.id_produs = :new.id_produs;
1074
1075     dbms_output.put_line('Aces produs se afla doar in meniul restaurantului ' ||
1076 EXCEPTION
1077   WHEN no_data_found THEN
1078     raise_application_error(-20065, 'Acest produs nu exista in niciun meniu')
1079   WHEN too_many_rows THEN
1080     dbms_output.put_line('Aces produs se afla in meniurile mai multor restaurante');
1081   END BEFORE EACH ROW;
1082   AFTER EACH ROW IS BEGIN
1083     dbms_output.put_line('Produsul cu id ' ||
1084                           || :new.id_produs
1085                           || ' in comanda');
1086   END AFTER EACH ROW;

```

Script Output x Query Result x  
✖ ✚ ✖ ✖ Task completed in 0.057 seconds

```

ORA-20067: Astazi nu se pot plasa comenzi, este zi libera!
ORA-06512: at "ROBERTTO.TRIG_VACANTA", line 6
ORA-04088: error during execution of trigger 'ROBERTTO.TRIG_VACANTA'

Trigger TRIG_PRODUS_IN_MENUI compiled

```

## 11.1 Declansarea acestuia:

The screenshot shows the Oracle SQL Developer interface with a query editor and an output window.

**Query Editor (Worksheet tab):**

```
1077 WHEN no_data_found THEN
1078     raise_application_error(-20065, 'Acest produs nu exista in niciun meniu');
1079 WHEN too_many_rows THEN
1080     dbms_output.put_line('Aces produs se afla in meniurile mai multor restaurante');
1081 END BEFORE EACH ROW;
1082 AFTER EACH ROW IS BEGIN
1083     dbms_output.put_line('Produsul cu id '
1084     || :new.id_produs
1085     || ' in comanda');
1086 END AFTER EACH ROW;
1087 AFTER STATEMENT IS BEGIN
1088     dbms_output.put_line('Verificare completa. Pofta buna!');
1089 END AFTER STATEMENT;
1090 END trig_produs_in_menu;
1091 /
1092
1093 insert into continut_comanda (id_produs, id_comanda, numar_produse)
1094 values (6, 4, 2); --nu se poate insera un produs care nu apare in niciun meniu
1095
1096 delete from continut_comanda
1097 where id_produs = 6 and id_comanda = 4;
1098
1099 insert into continut_comanda (id_produs, id_comanda, numar_produse)
1100 values (1, 6, 2); --in meniul carui restaurant apare produsul (doar in restaurant 4)
1101
1102 delete from continut_comanda
1103 where id_produs = 1 and id_comanda = 6;
1104
1105 insert into continut_comanda (id_produs, id_comanda, numar_produse)
1106 values (4, 3, 2); --produs care se afla in mai multe meniuri
1107
```

**Output Window (DML Output tab):**

```
Verificare daca produsul exista in meniu
insert into continut_comanda (id_produs, id_comanda, numar_produse)
values (6, 4, 2)
Error report -
ORA-20065: Acest produs nu exista in niciun meniu
ORA-06512: at "ROBERTTO.TRIG_PRODUS_IN_MENU", line 18
ORA-04088: error during execution of trigger 'ROBERTTO.TRIG_PRODUS_IN_MENU'
```

Worksheet | Query Builder

```

1077 WHEN no_data_found THEN
1078     raise_application_error(-20065, 'Acest produs nu exista in niciun meniu');
1079 WHEN too_many_rows THEN
1080     dbms_output.put_line('Aces produs se afla in meniurile mai multor restaurante');
1081 END BEFORE EACH ROW;
1082 AFTER EACH ROW IS BEGIN
1083     dbms_output.put_line('Produsul cu id '
1084                         || :new.id_produs
1085                         || ' in comanda');
1086 END AFTER EACH ROW;
1087 AFTER STATEMENT IS BEGIN
1088     dbms_output.put_line('Verificare completa. Pofta buna!');
1089 END AFTER STATEMENT;
1090 END trig_produs_in_menui;
1091 /
1092
1093 insert into continut_comanda (id_produs, id_comanda, numar_produse)
1094 values (6, 4, 2); --nu se poate insera un produs care nu apare in niciun meniu
1095
1096 delete from continut_comanda
1097 where id_produs = 6 and id_comanda = 4;
1098
1099 insert into continut_comanda (id_produs, id_comanda, numar_produse)
1100 values (1, 6, 2); --in meniul carui restaurant apare produsul (doar in restaurant 4)
1101
1102 delete from continut_comanda
1103 where id_produs = 1 and id_comanda = 6;
1104
1105 insert into continut_comanda (id_produs, id_comanda, numar_produse)
1106 values (4, 3, 2); --produs care se afla in mai multe meniuri
1107

```

Script Output | Query Result | Task completed in 0.071 seconds

1 row deleted.

1 row inserted.

PROJECT BD ROBERTO | Doms Output | Buffer Size:20000

Verificare daca produs exista in meniu  
Aces produs se afla doar in meniul restaurantului 4  
Produsul cu id 1 in comanda  
Verificare completa. Pofta buna!

Worksheet | Query Builder

```

1080     dbms_output.put_line('Aces produs se afla in meniurile mai multor restaurante');
1081 END BEFORE EACH ROW;
1082 AFTER EACH ROW IS BEGIN
1083     dbms_output.put_line('Produsul cu id '
1084                         || :new.id_produs
1085                         || ' in comanda');
1086 END AFTER EACH ROW;
1087 AFTER STATEMENT IS BEGIN
1088     dbms_output.put_line('Verificare completa. Pofta buna!');
1089 END AFTER STATEMENT;
1090 END trig_produs_in_menui;
1091 /
1092
1093 insert into continut_comanda (id_produs, id_comanda, numar_produse)
1094 values (6, 4, 2); --nu se poate insera un produs care nu apare in niciun meniu
1095
1096 delete from continut_comanda
1097 where id_produs = 6 and id_comanda = 4;
1098
1099 insert into continut_comanda (id_produs, id_comanda, numar_produse)
1100 values (1, 6, 2); --in meniul carui restaurant apare produsul (doar in restaurant 4)
1101
1102 delete from continut_comanda
1103 where id_produs = 1 and id_comanda = 6;
1104
1105 insert into continut_comanda (id_produs, id_comanda, numar_produse)
1106 values (4, 3, 2); --produs care se afla in mai multe meniuri
1107
1108 delete from continut_comanda
1109 where id_produs = 4 and id_comanda = 3;
1110

```

Script Output | Query Result | Task completed in 0.083 seconds

1 row deleted.

1 row inserted.

PROJECT BD ROBERTO | Doms Output | Buffer Size:20000

Verificare daca produs exista in meniu  
Aces produs se afla in meniurile mai multor restaurante  
Produsul cu id 4 in comanda  
Verificare completa. Pofta buna!

## Aici am mai făcut încă un Trigger:

Un client nu poate aştepta mai mult decât 300 minute pentru ca comanda să fie preparată. Dacă timpul de preparare 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:

The screenshot shows the Oracle SQL Developer interface with the SQL worksheet tab selected. The code for the trigger is displayed in the worksheet. The code is as follows:

```
1110
1111 CREATE OR REPLACE TRIGGER trig_timp_prep BEFORE
1112     INSERT OR UPDATE OF durata ON preparare
1113     FOR EACH ROW
1114 BEGIN
1115     IF :new.durata > 300 THEN
1116         raise_application_error(-20086, 'timp de preparare prea mare!');
1117     END IF;
1118 END;
1119 /
1120
1121 insert into preparare (id_produs, id_comanda, id_angajat, durata)
1122 values (1, 1, 17, 600);
1123
1124 -- Exercitiul 12
1125 CREATE OR REPLACE TRIGGER trig_ldd_client BEFORE DROP ON SCHEMA BEGIN
1126     IF lower(ora_dict_obj_name) = lower('Client') THEN
1127         raise_application_error(-20777, 'Nu aveti voie sa stergeti tabela cu clienti');
1128     END IF;
1129 END;
1130 /
1131
```

The code is numbered from 1110 to 1131. The trigger body (lines 1114-1119) is highlighted in yellow. The screenshot also shows the bottom of the interface with tabs for 'Script Output' and 'Query Result', and a status bar indicating 'Task completed in 0.047 seconds'.

1 row inserted.

Trigger TRIG\_TIMP\_PREP compiled

## L-am declarat:

```
insert into preparare (id_produs, id_comanda, id_angajat, durata)
values (1, 1, 17, 600);
```

### Print-Screen:

The screenshot shows the Oracle SQL Developer interface. In the top pane, there is a code editor with the following SQL code:

```
-- Exercitiul 12
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;
/
insert into preparare (id_produs, id_comanda, id_angajat, durata)
values (1, 1, 17, 600);
```

In the bottom pane, the "Script Output" tab is selected, showing the execution results:

```
Task completed in 0.044 seconds
insert into preparare (id_produs, id_comanda, id_angajat, durata)
values (1, 1, 17, 600)
Error report -
ORA-20086: timp de preparare prea mare!
ORA-06512: at "ROBERTTO.TRIG_TIMP_PREP", line 3
ORA-04088: error during execution of trigger 'ROBERTTO.TRIG_TIMP_PREP'
```

## 12. 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:

The screenshot shows the Oracle SQL Developer interface. In the top pane, there is a code editor with the following SQL code:

```
-- Exercitiul 12
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;
/
drop table client;
--SELECT * from preparare;
```

In the bottom pane, the "Script Output" tab is selected, showing the execution results:

```
Task completed in 0.061 seconds
ORA-20086: timp de preparare prea mare!
ORA-06512: at "ROBERTTO.TRIG_TIMP_PREP", line 3
ORA-04088: error during execution of trigger 'ROBERTTO.TRIG_TIMP_PREP'

Trigger TRIG_LDD_CLIENT compiled
```

## 12.1 Declansarea acestuia:

```
drop table client;
```

### Print-Screen:

The screenshot shows a SQL developer interface. In the script editor, line 1132 contains the command `drop table client;`. The output window displays an error stack starting with ORA-04088, indicating a trigger execution failure.

```
1131
1132 drop table client;
1133
1134 --SELECT * from preparare;
ORA-04088: error during execution of trigger 'ROBERTTO.TRIG_LDD_CLIENT'
ORA-00604: error occurred at recursive SQL level 1
ORA-20777: Nu aveti voie sa stergeti tabela cu clienti
ORA-06512: at line 3
04088. 00000 - "error during execution of trigger '%s.%s'"
*Cause: A runtime error occurred during execution of a trigger.
*Action: Check the triggers which were involved in the operation.
```

## 13. Crearea unui pachet care conține obiectele de la punctele 6-9:

```
CREATE OR REPLACE PACKAGE pachet_cerinte AS
-- TYPE tab_imb IS TABLE OF NUMBER(10);
-- Exercitiul 6
FUNCTION afis_produse_preferate (my_id_client IN client.id_client%TYPE) RETURN tab_imb;
-- Exercitiul 7
PROCEDURE afis_durata_preparare_produs;
-- Exercitiul 8
FUNCTION profit_client (my_id_client IN client.id_client%TYPE) RETURN PLS_INTEGER;
-- Exercitiul 9
PROCEDURE mai_multe_ingrediente_producator(my_id_comanda IN comanda.id_comanda%TYPE,
my_id_produs IN produs.id_produs%TYPE, my_id_producator IN producator.id_producator%TYPE,
my_numar_produse_comanda_continut_comanda.numar_produse%TYPE);
PROCEDURE afis_cantitate_ingredient_producator_produs(my_id_producator IN
producator.id_producator%TYPE);
END pachet_cerinte;
/
```

```

CREATE OR REPLACE PACKAGE BODY pachet_cerinte AS

    TYPE tab_imb IS TABLE OF NUMBER(10);

    FUNCTION afis_produse_preferate (my_id_client IN client.id_client%TYPE)
        RETURN tab_imb IS

        TYPE tab_ind IS TABLE OF PLS_INTEGER INDEX BY PLS_INTEGER;

        v_contor          tab_ind;
        v_index_max       PLS_INTEGER;
        id               PLS_INTEGER;
        v_id_feluri_preferate tab_imb;
        ok              BINARY_INTEGER := 0;

    BEGIN

        v_id_feluri_preferate := tab_imb();

        FOR i IN (SELECT p.id_produs -- CURSOR IMPLICIT
                  FROM produs p, continut_comanda cont, comanda cmd
                  WHERE my_id_client = cmd.id_client
                    AND cmd.id_comanda = cont.id_comanda
                    AND cont.id_produs = p.id_produs) LOOP BEGIN

            v_contor(i.id_produs) := v_contor(i.id_produs) + 1; -- 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) := 1; -- 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

DBMS_OUTPUT.PUT_LINE('Acest client nu a comandat nimic in viata lui');

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;

RETURN v_id_feluri_preferate;

END afis_produse_preferate;

```

```

PROCEDURE afis_durata_preparare_produs

IS TYPE refcursor IS REF CURSOR;

CURSOR c IS SELECT a.id_angajat, a.nume, a.prenume, b.nr_stele,

CURSOR (SELECT prep.durata, p.nume, p.descriere, p.id_produs

FROM preparare prep, produs p

WHERE b.id_angajat = prep.id_angajat

AND p.id_produs = prep.id_produs)

FROM angajat a, bucatar b

```

```

        WHERE a.id_angajat = b.id_angajat;

TYPE rec IS RECORD (id_angajat.id_angajat%TYPE, dur_preparare.durata%TYPE);

TYPE tab_ind IS TABLE OF rec INDEX BY PLS_INTEGER;

t          tab_ind;

v_cursor      refcursor;

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;

v_contor         PLS_INTEGER := 0;

v_nume_bucatar_best angajat.nume%TYPE;

v_prenume_bucatar_best angajat.prenume%TYPE;

v_nr_stele_bucatar_best bucatar.nr_stele%TYPE;

BEGIN

OPEN c;

LOOP

FETCH c INTO v_id_bucatar, v_nume_bucatar, v_prenume_bucatar, v_nr_stele_bucatar, v_cursor;

EXIT WHEN c%NOTFOUND;

FETCH v_cursor INTO v_durata_preparare, v_nume_produs, v_descriere_produs, v_id_produs;

IF v_cursor%NOTFOUND THEN

    DBMS_OUTPUT.PUT_LINE('ATENTIE! Bucatarul ' || v_nume_bucatar || ' ' ||
v_prenume_bucatar || ' nu a preparat niciodata un produs!');

    DBMS_OUTPUT.NEW_LINE();

ELSE

```

```

DBMS_OUTPUT.PUT_LINE('Bucatarul ' || v_nume_bucatar || ' ' || v_prenume_bucatar || ' a
preparat urmatoarele produse:');
LOOP
EXIT WHEN v_cursor%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(v_cursor%ROWCOUNT || '. produsul: ' || v_nume_produs || ' cu
descrierea ' || 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 v_cursor INTO v_durata_preparare, v_nume_produs, v_descriere_produs,
v_id_produs;
END LOOP;
END IF;
END LOOP;
CLOSE c;
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.');

```

```

END LOOP;

END afis_durata_preparare_produs;

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;
v_exista             PLS_INTEGER := 0;
t                    tab_ind;

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

            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;

RETURN t(my_id_client);

EXCEPTION
    WHEN invalid THEN
        RAISE_APPLICATION_ERROR(-20021, 'Nu a fost gasit niciun client cu id-ul introdus!');

    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20022, 'Alta eroare!');

END profit_client;

```

PROCEDURE mai\_multe\_ingrediente\_producator(my\_id\_comanda IN comanda.id\_comanda%TYPE,  
my\_id\_produs IN produs.id\_produs%TYPE, my\_id\_producator IN producator.id\_producator%TYPE,  
my\_numar\_produse\_comanda\_continut\_comanda.numar\_produse%TYPE)

```

IS
    TYPE rec IS RECORD (v_cantitate_produs cantitate_produs.cantitate%TYPE, v_nume_ingredient
ingredient.nume%TYPE);

    TYPE tab_ind IS TABLE OF rec INDEX BY PLS_INTEGER;
    t_tab_ind;

BEGIN
    SELECT cnt.cantitate, i.nume BULK COLLECT INTO t
    FROM cantitate_produs cnt, ingredient i, producator prod, produs p, continut_comanda cmd
    WHERE cmd.id_comanda = my_id_comanda
    AND cmd.id_produs = my_id_produs
    AND p.id_produs = cmd.id_produs
    AND cnt.id_produs = p.id_produs
    AND i.id_ingredient = cnt.id_ingredient
    AND prod.id_producator = i.id_producator
    AND my_id_producator = prod.id_producator;
    FOR i IN 1..t.LAST LOOP

```

```

        DBMS_OUTPUT.PUT_LINE('-> ' || my_numar_produse_comanda*t(i).v_cantitate_produs || ' de
' || t(i).v_nume_ingredient || '.');
    END LOOP;

END mai_multe_ingrediente_producator;

PROCEDURE afis_cantitate_ingredient_producator_produs(my_id_producator IN
producator.id_producator%TYPE) IS
    CURSOR mycursor IS (SELECT p.id_produs, p.nume, cmd.numar_produse, cmd.id_comanda
                        FROM produs p, continut_comanda cmd
                        WHERE p.id_produs=cmd.id_produs);
    TYPE tab_ind IS TABLE OF ingredient.nume%TYPE INDEX BY PLS_INTEGER;
    v_id_produs          produs.id_produs%TYPE;
    v_id_comanda         continut_comanda.id_comanda%TYPE;
    v_nume_produs        produs.nume%TYPE;
    v_numar_produse      continut_comanda.numar_produse%TYPE;
    v_cantitate_produs   cantitate_produs.cantitate%TYPE;
    v_nume_ingredient     ingredient.nume%TYPE;
    v_nume_producator    producator.nume%TYPE;
    v_nr_telefon_producator  producator.nr_telefon%TYPE;
    lista_ingrediente_producator  tab_ind;
BEGIN
    SELECT nume, nr_telefon INTO v_nume_producator, v_nr_telefon_producator
    FROM producator
    WHERE id_producator = my_id_producator;
    SELECT nume BULK COLLECT INTO lista_ingrediente_producator
    FROM ingredient
    WHERE id_producator = my_id_producator;
    DBMS_OUTPUT.PUT_LINE('Numele producatorului selectat este ' || v_nume_producator || '.');
    DBMS_OUTPUT.PUT_LINE('El poate fi contactat la numarul de telefon: ' ||
v_nr_telefon_producator || '.');

```

```

DBMS_OUTPUT.PUT_LINE('Acesta produce urmatoarele ingrediente:');
FOR i IN 1..lista_ingrediente_producator.LAST LOOP
    DBMS_OUTPUT.PUT_LINE('-> ' || lista_ingrediente_producator(i) || ':');
END LOOP;
DBMS_OUTPUT.NEW_LINE();
OPEN mycursor;
LOOP
    FETCH mycursor INTO v_id_produs,v_nume_produs, v_numar_produse, v_id_comanda;
    EXIT WHEN mycursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('-- ' || v_nume_produs || ' are de la producatorul ' ||
v_nume_producator || ' in comanda cu id-ul ' || v_id_comanda || ' urmatorul ingredient:');
    BEGIN
        SELECT cnt.cantitate, i.nume INTO v_cantitate_produs, v_nume_ingredient
        FROM cantitate_produs cnt, ingredient i, producator prod, produs p, continut_comanda cmd
        WHERE cmd.id_comanda = v_id_comanda
        AND cmd.id_produs = v_id_produs
        AND p.id_produs = cmd.id_produs
        AND cnt.id_produs = p.id_produs
        AND i.id_ingredient = cnt.id_ingredient
        AND prod.id_producator = i.id_producator
        AND my_id_producator = prod.id_producator;
        DBMS_OUTPUT.PUT_LINE('-> ' || v_numar_produse*v_cantitate_produs || 'g de ' ||
v_nume_ingredient || '.');
        DBMS_OUTPUT.NEW_LINE();
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('ATENTIE! Nu exista niciun ingredient al acestui producator in
acest preparat!');
            DBMS_OUTPUT.NEW_LINE();
        WHEN TOO_MANY_ROWS THEN

```

```

        DBMS_OUTPUT.PUT_LINE('ATENTIE! Mai multe ingrediente ale acestui producator detectate
in acest preparat!');

        BEGIN

            mai_multe_ingrediente_producator(v_id_comanda, v_id_produs, my_id_producator,
v_numar_produse);

        END;

        DBMS_OUTPUT.NEW_LINE();

    END;

    END LOOP;

    CLOSE mycursor;

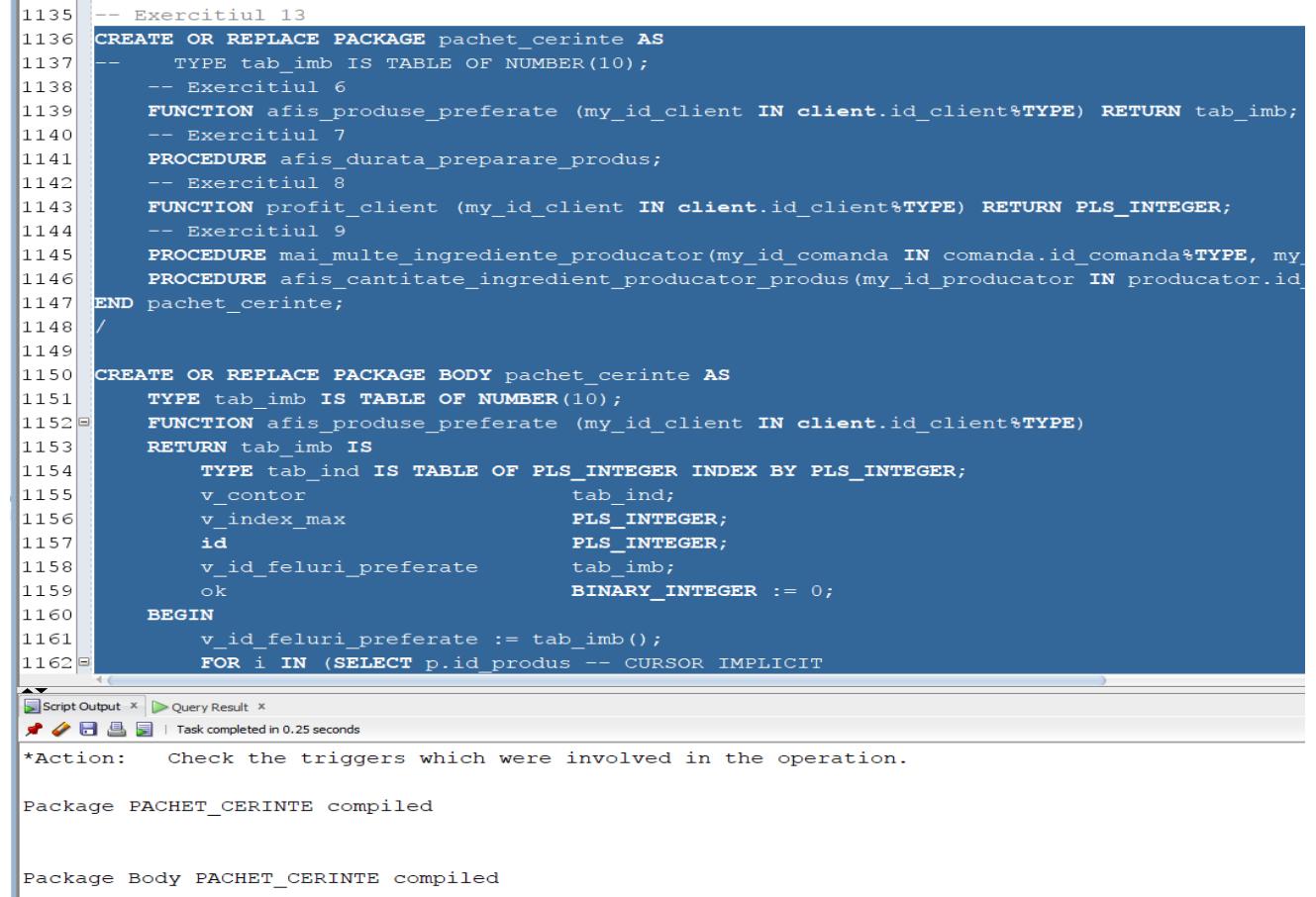
END afis_cantitate_ingredient_producator_produs;

END pachet_cerinte;

```

/

***Print-Screen:***



```

1135 -- Exercitiul 13
1136 CREATE OR REPLACE PACKAGE pachet_cerinte AS
1137     -- TYPE tab_imb IS TABLE OF NUMBER(10);
1138     -- Exercitiul 6
1139     FUNCTION afis_produse_preferate (my_id_client IN client.id_client%TYPE) RETURN tab_imb;
1140     -- Exercitiul 7
1141     PROCEDURE afis_durata_preparare_produs;
1142     -- Exercitiul 8
1143     FUNCTION profit_client (my_id_client IN client.id_client%TYPE) RETURN PLS_INTEGER;
1144     -- Exercitiul 9
1145     PROCEDURE mai_multe_ingrediente_producator (my_id_comanda IN comanda.id_comanda%TYPE, my_id_producator IN producator.id_producator%TYPE);
1146     PROCEDURE afis_cantitate_ingredient_producator_produs (my_id_producator IN producator.id_producator%TYPE);
1147 END pachet_cerinte;
1148 /
1149
1150 CREATE OR REPLACE PACKAGE BODY pachet_cerinte AS
1151     TYPE tab_imb IS TABLE OF NUMBER(10);
1152     FUNCTION afis_produse_preferate (my_id_client IN client.id_client%TYPE)
1153         RETURN tab_imb IS
1154         TYPE tab_ind IS TABLE OF PLS_INTEGER INDEX BY PLS_INTEGER;
1155         v_contor          tab_ind;
1156         v_index_max       PLS_INTEGER;
1157         id                PLS_INTEGER;
1158         v_id_feluri_preferate tab_imb;
1159         ok               BINARY_INTEGER := 0;
1160     BEGIN
1161         v_id_feluri_preferate := tab_imb();
1162         FOR i IN (SELECT p.id_produs -- CURSOR IMPLICIT

```

Script Output | Query Result | Task completed in 0.25 seconds

\*Action: Check the triggers which were involved in the operation.

Package PACHET\_CERINTE compiled

Package Body PACHET\_CERINTE compiled

## **Apelare din pachet:**

-- Exercitiul 6  
-- Produsele preferate pentru un client citit de la tastatura  
-- Se afiseaza numele si prenumele clientului, produsele preferate si de cate ori le-a comandat.

```
DECLARE
    v_id_client_citit    client.id_client%TYPE := &client_id;
    my_tab                 tab_imb;
    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
    my_tab := pachet_cerinte.afis_produse_preferate(v_id_client_citit);
    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;

END;

```

```

1407 /
1408
1409 -- Exercitiul 6
1410 -- Produsele preferate pentru un client citit de la tastatura
1411 -- Se afiseaza numele si prenumele clientului, produsele preferate si de cate ori le-a comandat.
1412
1413 DECLARE
1414   v_id_client_citit      client.id_client%TYPE := &client_id;
1415   my_tab                  tab_imb;
1416   id                      PLS_INTEGER;
1417   v_nume_produs          produs.nume%TYPE;
1418   v_name                  client.nume%TYPE;
1419   v_prenume               client.prenume%TYPE;
1420   v_contor                PLS_INTEGER := 0;
1421 BEGIN
1422   my_tab := pachet_cerinte.afis_produse_preferate(v_id_client_citit);
1423   SELECT c.nume, c.prenume INTO v_name, v_prenume
1424   FROM client c
1425   WHERE c.id_client = v_id_client_citit;
1426   DBMS_OUTPUT.PUT_LINE('Clientul ' || v_name || ' ' || v_prenume || ' are urmatoarele produse favorite:');
1427   id := my_tab.FIRST; -- prima valoare din tabelul indexat (vector frecventa)
1428   LOOP
1429     EXIT WHEN id IS NULL;
1430     SELECT p.nume INTO v_nume_produs
1431     FROM produs p
1432     WHERE p.id_produs = my_tab(id);
1433     -- afisam produsele cu numarul maxim de aparitii
1434     v_contor := v_contor + 1;
1435     DBMS_OUTPUT.PUT_LINE(v_contor || '. ' || v_nume_produs);
1436     id := my_tab.NEXT(id);
1437   END LOOP;
1438
1439   DBMS_OUTPUT.PUT_LINE(v_contor || '. ' || v_nume_produs);
1440   id := my_tab.NEXT(id);
1441   END LOOP;
1442
1443 END;

```

PL/SQL procedure successfully completed.

### -- Exercitiul 7

-- Sa se afiseze pentru fiecare bucatar numele si prenumele acestuia, daca a preparat vreun produs de cand s-a angajat.

-- Daca a preparat minim un produs se va afisa date corespunzatoare legate de acesta (numele produsului, descrierea acestuia si

-- cat a durat sa il prepare).

```
BEGIN
```

```
  pachet_cerinte.afis_durata_preparare_produs();
```

```
END;
```

```
/
```

```

PROJECT BD ROBERTO
Query Builder
1428    LOOP
1429        EXIT WHEN id IS NULL;
1430        SELECT p.nume INTO v_nume_produs
1431        FROM produs p
1432        WHERE p.id_produs = my_tab(id);
1433        -- afisam produsele cu numarul maxim de aparitii
1434        v_contor := v_contor + 1;
1435        DBMS_OUTPUT.PUT_LINE(v_contor || ' ' || v_nume_produs);
1436        id := my_tab.NEXT(id);
1437    END LOOP;
1438
1439-- Exercitiul 7
1440-- Sa se afiseze pentru fiecare bucatar numele si prenumele acestuia, daca a preparat vreun produs de cand s-a angajat
1441-- Daca a preparat minim un produs se va afisa date corespunzatoare legate de acesta (numele produsului, descrierea
1442-- cat a durat sa il prepare).
1443
1444BEGIN
1445    pachet_cerinte.afis_durata_preparare_produs();
1446END;
1447/
1448
1449-- Exercitiul 8
1450-- Sa se afiseze profitul generat de un client citit de la tastatura, de-a lungul timpului,
1451-- Daca aceasta a beneficiat de reduceri de-a lungul timpului se va afisa valoarea totala a acestor reduceri.
1452-- Daca profitul adus de-a lungul timpului este unul negativ inseamna ca clientul a beneficiat de prea multe reduceri
1453-- si se va afisa un mesaj corespunzator.
1454-- De asemenea se va afisa profitul maxim oferit de un client si profitul obtinut de companie de-a lungul timpului.
1455
1456DECLARE
1457    v_id_client_citit    client.id_client%TYPE := &client_id;
1458
1459    v_rezultat      PLS_INTEGER;
1460
1461    BEGIN
1462        v_rezultat := 0;
1463        FOR i IN 1..v_id_client_citit
1464        LOOP
1465            v_rezultat := v_rezultat + v_id_client_citit;
1466        END LOOP;
1467        DBMS_OUTPUT.PUT_LINE('Profitul total este: ' || v_rezultat);
1468    END;
1469
1470-- ATENTIE! Bucatarul Daniel Andrei nu a preparat niciodata un produs!
1471
1472-- Bucatarul Filip Mihnea a preparat produsul Spaghete in 45 minute.
1473-- Bucatarul Scarlatescu Catalin a preparat produsul Pizza in 35 minute.
1474-- Bucatarul Filip Mihnea a preparat produsul Paste bolognese in 16 minute.
1475-- Bucatarul Scarlatescu Catalin a preparat produsul Sarmale in 120 minute.
1476-- Bucatarul Dumitrescu Florin a preparat produsul Ciorba de post in 75 minute.
1477
1478END;

```

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

## -- Exercitiul 8

-- Sa se afiseze profitul generat de un client citit de la tastatura, de-a lungul timpului,

-- Daca aceasta a beneficiat de reduceri de-a lungul timpului se va afisa valoarea totala a acestor reduceri.

-- Daca profitul adus de-a lungul timpului este unul negativ inseamna ca clientul a beneficiat de prea multe reduceri

-- si se va afisa un mesaj corespunzator.

-- De asemenea se va afisa profitul maxim oferit de un client si profitul obtinut de companie de-a lungul timpului.

**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;
/

```

The screenshot shows the Oracle SQL Developer interface. On the left, the 'Worksheet' tab is active, displaying a PL/SQL block. On the right, the 'DMS Output' tab is active, showing the results of the execution.

```

1443 -- cat a durat sa il prepare.
1444
1445 BEGIN
1446     pachet_cerinte.afis_durata_preparare_produs();
1447 END;
1448 /
1449
1450 -- Exercitiul 8
1451 -- Sa se afiseze profitul generat de un client citit de la tastatura, de-a lungul timpului,
1452 -- Daca acesta a beneficiat de reduceri de-a lungul timpului se va afisa valoarea totala a acestor reduceri.
1453 -- Daca profitul adus de-a lungul timpului este unul negativ inseamna ca clientul a beneficiat de prea multe reduceri
1454 -- si se va afisa un mesaj corespunzator.
1455 -- De asemenea se va afisa profitul maxim oferit de un client si profitul obtinut de companie de-a lungul timpului.
1456
1457 DECLARE
1458     v_id_client_citit    client.id_client%TYPE := &client_id;
1459     v_rezultat          PLS_INTEGER;
1460
1461 BEGIN
1462     v_rezultat := pachet_cerinte.profit_client(v_id_client_citit);
1463     IF v_rezultat > 0 THEN
1464         DBMS_OUTPUT.PUT_LINE('El a contribuit la profitul companiei pana acum cu ' || v_rezultat || ' lei.');
1465     ELSE
1466         DBMS_OUTPUT.PUT_LINE('Acesta a beneficiat de prea multe reduceri! Aducand un profit negativ companiei: ' || v_rezultat || ' lei.');
1467     END IF;
1468 END;
1469 /
1470

```

DMS Output window content:

```

PROJECT BD ROBERTO *
DMS Output
Buffer Size:20000
PROJECT BD ROBERTO *
0.92199999 seconds
PROJECT BD ROBERTO *
DMS Output
Buffer Size:20000
PROJECT BD ROBERTO *
Profital obtinut in urma clientilor de pana acum este de 80 lei.
Profitul maxim adus de un client este de 30 lei.
Clientul cu id-ul 3 este Gheorghe Sebastian cu nr de telefon: 0744573419.
Nu beneficiaza de reduceri!
El a contribuit la profitul companiei pana acum cu 20 lei.

```

### -- Exercitiul 9

-- Sa se afiseze ce cantitate ingrediente contine fiecare produs, din fiecare comanda,  
-- furnizate de un producator citit de la tastatura.

```

DECLARE
    v_id_producator producator.id_producator%TYPE := &producator_id;

```

BEGIN

```
afis_cantitate_ingredient_producator_produs(v_id_producator);
```

END;

/

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there is a 'Query Builder' window with some commented-out code. The main 'Worksheet' tab contains a PL/SQL block:

```
BEGIN
    afis_cantitate_ingredient_producator_produs(v_id_producator);
END;
/

```

The 'Script Output' tab at the bottom shows the execution results:

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

The right side of the interface shows the 'PROJECT BD ROBERTTO' database structure with tables like 'bors', 'parmezan', and 'cascaval'. Below the database structure, there are several comments about ingredients:

- Spaghete are de la producatorul Olivers in comanda cu id-ul 4 urmatorul ingredient: ATENTIE! Nu exista niciun ingredient al acestui producator in acest preparat!
- Spaghete are de la producatorul Olivers in comanda cu id-ul 2 urmatorul ingredient: ATENTIE! Nu exista niciun ingredient al acestui producator in acest preparat!
- Pizza are de la producatorul Olivers in comanda cu id-ul 2 urmatorul ingredient: ATENTIE! Mai multe ingrediente ale acestui producator detectate in acest preparat!
- > 50g de parmezan.
- > 35g de cascaval.
- Pizza are de la producatorul Olivers in comanda cu id-ul 1 urmatorul ingredient: ATENTIE! Mai multe ingrediente ale acestui producator detectate in acest preparat!
- > 50g de parmezan.
- > 35g de cascaval.
- Paste bolognese are de la producatorul Olivers in comanda cu id-ul 5 urmatorul ingredient: ATENTIE!
- > 55g de bors.
- Paste bolognese are de la producatorul Olivers in comanda cu id-ul 4 urmatorul ingredient: ATENTIE!
- > 55g de bors.
- Sarmale are de la producatorul Olivers in comanda cu id-ul 5 urmatorul ingredient: ATENTIE! Nu exista niciun ingredient al acestui producator in acest preparat!
- Ciorba de post are de la producatorul Olivers in comanda cu id-ul 4 urmatorul ingredient: ATENTIE!
- > 350g de bors.
- Ciorba de post are de la producatorul Olivers in comanda cu id-ul 2 urmatorul ingredient: ATENTIE!
- > 350g de bors.
- Paste bolognese are de la producatorul Olivers in comanda cu id-ul 3 urmatorul ingredient: ATENTIE!
- > 55g de bors.
- Pizza are de la producatorul Olivers in comanda cu id-ul 6 urmatorul ingredient: ATENTIE! Mai multe ingrediente ale acestui producator detectate in acest preparat!
- > 100g de parmezan.
- > 70g de cascaval.
- Sarmale are de la producatorul Olivers in comanda cu id-ul 3 urmatorul ingredient: ATENTIE! Nu exista niciun ingredient al acestui producator in acest preparat!

## 14. Crearea unui pachet care conține tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specific bazei de date:

Acest pachet ușurează munca administratorului bazei de date, întrucât oferă informații detaliate despre clienți și ofertele acestora.

CREATE OR REPLACE PACKAGE pachet\_robertto AS

```
TYPE rec IS RECORD (v_id_com comanda.id_comanda%TYPE, v_pret_com comanda.pret%TYPE,
v_id_client comanda.id_client%TYPE);
```

```

TYPE t_comenzi IS VARRAY(100) OF rec;
comenzi  t_comenzi:= t_comenzi();
FUNCTION gaseste_client_pret_preferential(my_client_id client.id_client%TYPE) RETURN BOOLEAN;
FUNCTION afis_comenzi_pref_zi(my_ziua DATE) RETURN t_comenzi;
PROCEDURE sortare(v IN OUT t_comenzi);
PROCEDURE afis_detalii_comenzi(comenzi t_comenzi);
PROCEDURE f_pachet_robertto;
END pachet_robertto;
/

```

CREATE OR REPLACE PACKAGE BODY pachet\_robertto

IS

-- Verifica daca un client mai poate beneficia de o reducere pentru comanda sa

```

FUNCTION gaseste_client_pret_preferential(my_client_id client.id_client%TYPE)
RETURN BOOLEAN

```

```

IS gasit PLS_INTEGER := 0;
```

```

BEGIN
```

```

    SELECT 1 INTO gasit
```

```

        FROM client cli, comanda cmd, factura f
```

```

        WHERE my_client_id = cmd.id_client
```

```

        AND cmd.id_factura = f.id_factura
```

```

        AND f.detalii LIKE UPPER('%card%')
```

```

        AND EXTRACT (MONTH FROM cmd.data) IN (1, 5, 12)
```

```

        AND f.valoare > 80;
```

```

EXCEPTION
```

```

    WHEN NO_DATA_FOUND THEN
```

```

        RETURN FALSE;
```

```

    WHEN TOO_MANY_ROWS THEN
```

```
    RETURN TRUE;  
END gaseste_client_pret_preferential;
```

FUNCTION afis\_comenzi\_pref\_zi(my\_ziua DATE) RETURN t\_comenzi --afisaza comenzile dîtr-o zi ale clientilor care beneficiaza de pret preferential

```
IS  
comenzi_clienti_preferati t_comenzi := t_comenzi();
```

```
BEGIN  
SELECT c.id_comanda, c.pret, c.id_client  
BULK COLLECT INTO comenzi  
FROM comanda c  
WHERE to_char(c.data,'dd-mm-yyyy') = to_char(my_ziua,'dd-mm-yyyy');
```

```
FOR com IN 1..comenzi.count LOOP  
IF gaseste_client_pret_preferential(comenzi(com).v_id_client) = true THEN  
    DBMS_OUTPUT.PUT_LINE('Comanda ' || comenzi(com).v_id_com || ' a clientului cu id ' ||  
comenzi(com).v_id_client || ' are pretul ' || comenzi(com).v_pret_com);  
    comenzi_clienti_preferati.EXTEND();  
    comenzi_clienti_preferati(comenzi_clienti_preferati.COUNT):=comenzi(com);  
END IF;  
END LOOP;
```

```
RETURN comenzi_clienti_preferati;  
END afis_comenzi_pref_zi;
```

```
PROCEDURE sortare(v IN OUT t_comenzi)  
IS
```

```

ok BOOLEAN;
aux rec;
BEGIN
LOOP
    ok := false; -- False
    FOR i IN 2..v.count LOOP
        IF v(i - 1).v_pret_com > v(i).v_pret_com THEN
            -- facem swap
            aux := v(i);
            v(i) := v(i - 1);
            v(i - 1) := aux;
            ok := true;
        END IF;
    END LOOP;

    EXIT WHEN NOT ok; -- daca am parcurs sortat si nu am facut nicio interschimbare (swap)
    -- inseamna ca tabloul nostru este sortat
END LOOP;

FOR i IN 1..v.count LOOP
    dbms_output.put_line(v(i).v_id_com || ' ' || v(i).v_pret_com);
END LOOP;
END sortare;

PROCEDURE afis_detalii_comenzi(comenzi t_comenzi) IS
BEGIN
    FOR i IN 1..comenzi.count LOOP
        DBMS_OUTPUT.PUT_LINE('Comanda ' || comenzi(i).v_id_com || ' are urmatoarele produse:');
        FOR v_cursor IN (SELECT p.nume n, cnt.numar_produse nr

```

```

        FROM produs p, continut_comanda cnt, comanda cmd
        WHERE cmd.id_comanda = cnt.id_comanda
        AND cnt.id_produs = p.id_produs
        AND cmd.id_comanda=comenzi(i).v_id_com)

LOOP
    DBMS_OUTPUT.PUT_LINE(v_cursor.nr || '*' || v_cursor.n);
END LOOP;
DBMS_OUTPUT.PUT_LINE('.....');
END LOOP;

END afis_detalii_comenzi;

PROCEDURE f_pachet_robertto IS
v t_comenzi := t_comenzi();
BEGIN
v:=afis_comenzi_pref_zi(to_date('01-07-2022', 'mm-dd-yyyy'));
sortare(v);
afis_detalii_comenzi(v);
END f_pachet_robertto;
END pachet_robertto;
/

```

### **Print-Screen:**

```
1484 -- Exercitiul 14
1485 CREATE OR REPLACE PACKAGE pachet_robertto AS
1486     TYPE rec IS RECORD (v_id_com comanda.id_comanda%TYPE, v_pret_com comanda.pret%TYPE, v_id_client comanda.id_client%TYPE);
1487     TYPE t_comenzi IS VARRAY(100) OF rec;
1488     comenzi    t_comenzi:= t_comenzi();
1489     FUNCTION gaseste_client_pret_preferential(my_client_id client.id_client%TYPE) RETURN BOOLEAN;
1490     FUNCTION afis_comenzi_pref_zi(my_ziua DATE) RETURN t_comenzi;
1491     PROCEDURE sortare(v IN OUT t_comenzi);
1492     PROCEDURE afis_detalii_comenzi(comenzi t_comenzi);
1493     PROCEDURE f_pachet_robertto;
1494 END pachet_robertto;
1495 /
1496
1497 CREATE OR REPLACE PACKAGE BODY pachet_robertto
1498 IS
1499
1500     -- Verifica daca un client mai poate beneficia de o reducere pentru comanda sa
1501     FUNCTION gaseste_client_pret_preferential(my_client_id client.id_client%TYPE)
1502         RETURN BOOLEAN
1503     IS
1504         gasit PLS_INTEGER := 0;
1505     BEGIN
1506         SELECT 1 INTO gasit
1507             FROM client cli, comanda cmd, factura f
1508             WHERE my_client_id = cmd.id_client
1509                 AND cmd.id_factura = f.id_factura
1510                 AND f.detalii LIKE UPPER('%card%')
1511                 AND EXTRACT (MONTH FROM cmd.data) IN (1, 5, 12)
1512                 AND f.valoare > 80;
1513     EXCEPTION
1514         WHEN NO_DATA_FOUND THEN
1515             RETURN FALSE;
1516     
```

Script Output | Query Result | Task completed in 0.109 seconds

Package PACHET\_ROBERTTO compiled

Package Body PACHET\_ROBERTTO compiled

### **Apelare din pachet:**

-- Sa se verifice daca un client citit de la tastatura poate beneficia de o reducere pentru comenzile plasate de acesta

-- Criteriile pentru a putea beneficia de o reducere de 10% din pretul normal al comenzii sunt:

- -> sa fi achitat comanda respectiva folosind cardul
- -> luna in care a plasat comanda sa fie Ianurie, Mai, Decembrie
- -> sa aiba o valoare de peste 80 lei

DECLARE

```
v_id_client_citit client.id_client%TYPE := &client_id;
ok      BOOLEAN;
```

BEGIN

```
ok := pachet_robertto.gaseste_client_pret_preferential(v_id_client_citit);
```

IF ok THEN

```
DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul dat este eligibil pentru un pret promotional la momentul actual!');
```

ELSE

```
DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul dat NU este eligibil pentru un pret promotional la momentul actual!');
```

END IF;

END;

Pentru id\_client = 2:

```
PROJECT ED ROBERTTO
3.0999999 seconds
PROJECT ED ROBERTTO
Buffer Size:20000
PROJECT ED ROBERTTO x
Clientul cu id-ul dat NU este eligibil pentru un pret promotional la momentul actual!
Script Output 4 | Query Result x
Task completed in 3.04 seconds
+-----+
| Procedure f_pachet_robertto |
+-----+
1583 PROCEDURE f_pachet_robertto IS
1584   v_t_comenzi := t_comenzi();
1585   BEGIN
1586     v:=afis_comenzi_pref_zi(to_date('01-07-2022', 'mm-dd-yyyy'));
1587     sortare(v);
1588     afis_detalii_comenzi(v);
1589   END f_pachet_robertto;
1590 END pachet_robertto;
1591 /
1592
1593 select * from comanda;
1594
1595 -- Sa se verifice daca un client citit de la tastatura poate beneficia de o reducere pentru comenzile plasate de acesta
1596 -- Criteriile pentru a putea beneficia de o reducere de 10% din pretul normal al comenzii sunt:
1597 --      -> sa fi achitat comanda respectiva folosind cardul
1598 --      -> luna in care a plasat comanda sa fie Ianurie, Mai, Decembrie
1599 --      -> sa aiba o valoare de peste 80 lei
1600
1601 DECLARE
1602   v_id_client_citit  client.id_client%TYPE := &client_id;
1603   ok                 BOOLEAN;
1604 BEGIN
1605   ok := pachet_robertto.gaseste_client_pret_preferential(v_id_client_citit);
1606   IF ok THEN
1607     DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul dat este eligibil pentru un pret promotional la momentul actual!');
1608   ELSE
1609     DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul dat NU este eligibil pentru un pret promotional la momentul actual!');
1610   END IF;
1611 END;
1612
1613 DECLARE
1614
1615   ELSE
1616     DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul dat NU este eligibil pentru un pret promotional la momentul actual!');
1617   END IF;
1618
1619 END;
1620
1621
PL/SQL procedure successfully completed.
```

Pentru id\_client = 3:

The screenshot shows the Oracle SQL Developer interface with two panes. The left pane displays the PL/SQL code for a procedure named f\_pachet\_robertto. The right pane shows the output window where the procedure has been successfully executed.

```
1583 PROCEDURE f_pachet_robertto IS
1584     v_t_comenzi := t_comenzi();
1585     BEGIN
1586         v:=afis_comenzi_pref_zi(to_date('01-07-2022', 'mm-dd-yyyy'));
1587         sortare(v);
1588         afis_detalii_comenzi(v);
1589     END f_pachet_robertto;
1590 END pachet_robertto;
1591 /
1592
1593 select * from comanda;
1594
1595 -- Sa se verifice daca un client citit de la tastatura poate beneficia de o reducere pentru comenzile plasate de aceea
1596 -- Criteriile pentru a putea beneficia de o reducere de 10% din pretul normal al comenzii sunt:
1597 --     -> sa fi achitat comanda respectiva folosind cardul
1598 --     -> luna in care a plasat comanda sa fie Ianurie, Mai, Decembrie
1599 --     -> sa aiba o valoare de peste 80 lei
1600
1601 DECLARE
1602     v_id_client_citit    client.id_client%TYPE := &client_id;
1603     ok                  BOOLEAN;
1604 BEGIN
1605     ok := pachet_robertto.gaseste_client_pret_preferential(v_id_client_citit);
1606     IF ok THEN
1607         DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul dat este eligibil pentru un pret promotional la momentul actual!');
1608     ELSE
1609         DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul dat NU este eligibil pentru un pret promotional la momentul actual!');
1610     END IF;
1611 END;
1612
1613//DECLARE
```

Output pane:

```
Clientul cu id-ul dat este eligibil pentru un pret promotional la momentul actual!
```

Script Output pane:

```
Task completed in 0.746 seconds
```

Message:

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

```

DECLARE
    v_pachet_robertto.t_comenzi;

BEGIN
    v := pachet_robertto.afis_comenzi_pref_zi(sysdate);

END;
/

```

The screenshot shows the Oracle SQL Developer interface with two panes. The left pane is the 'Script' tab containing the PL/SQL code. The right pane is the 'Output' tab showing the results of the execution.

```

PROJECT BD ROBERTTO
0.039 seconds
1597 --    -> sa fi achitat comanda respectiva folosind cardul
1598 --    -> luna in care a plasat comanda sa fie Ianurie, Mai, Decembrie
1599 --    -> sa aiba o valoare de peste 80 lei
1600
1601 DECLARE
1602     v_id_client_citit    client.id_client%TYPE := &client_id;
1603     ok                      BOOLEAN;
1604 BEGIN
1605     ok := pachet_robertto.gaseste_client_pret_preferential(v_id_client_citit);
1606     IF ok THEN
1607         DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul dat este eligibil pentru un pret promotional la momentul actual!');
1608     ELSE
1609         DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul dat NU este eligibil pentru un pret promotional la momentul actual!');
1610     END IF;
1611 END;
1612
1613 DECLARE
1614     v_pachet_robertto.t_comenzi;
1615 BEGIN
1616     v := pachet_robertto.afis_comenzi_pref_zi(sysdate);
1617 END;
1618 /
1619
1620 DECLARE
1621     v_pachet_robertto.t_comenzi;
1622 BEGIN
1623     SELECT c.id_comanda, c.pret, c.id_client BULK COLLECT INTO v
1624     FROM comanda c;
1625
1626     pachet_robertto.sortare(v);
1627 END;

```

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

Comanda 2 a clientului cu id 3 are pretul 130  
Comanda 4 a clientului cu id 4 are pretul 230

```

DECLARE
    v_pachet_robertto.t_comenzi;

BEGIN
    SELECT c.id_comanda, c.pret, c.id_client BULK COLLECT INTO v
    FROM comanda c;

```

```

    pachet_robertto.sortare(v);

END;

/

```

The screenshot shows the Oracle SQL Developer interface with two main tabs visible:

- Worksheet**: Contains the PL/SQL code for the procedure. The code includes logic to check if a client is eligible for a promotional price based on their ID, and to sort commands by price. It also calls a function to execute the package body.
- Dbms Output**: Shows the results of the DBMS\_OUTPUT.PUT\_LINE statements, listing various client IDs and their corresponding promotional prices.

```

1603    ok          BOOLEAN;
1604  BEGIN
1605    ok := pachet_robertto.gaseste_client_pret_preferential(v_id_client_citit);
1606  IF ok THEN
1607    DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul dat este eligibil pentru un pret promotional la momentul actual!');
1608  ELSE
1609    DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul dat NU este eligibil pentru un pret promotional la momentul actual!');
1610  END IF;
1611 END;
1612
1613 DECLARE
1614   v pachet_robertto.t_comenzi;
1615 BEGIN
1616   v := pachet_robertto.afis_comenzi_pref_zi(sysdate);
1617 END;
1618 /
1619
1620 DECLARE
1621   v pachet_robertto.t_comenzi;
1622 BEGIN
1623   SELECT c.id_comanda, c.pret, c.id_client BULK COLLECT INTO v
1624   FROM comanda c;
1625
1626   pachet_robertto.sortare(v);
1627 END;
1628 /
1629
1630 BEGIN
1631   pachet_robertto.f_pachet_robertto();
1632 END;
1633 /

```

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

	Value
3	20
1	50
8	50
5	80
6	100
2	130
4	230

BEGIN

```
pachet_robertto.f_pachet_robertto();
```

END;

/

The screenshot shows the Oracle SQL Developer interface with a PL/SQL procedure being run. The code in the worksheet is:

```
1603    ok      BOOLEAN;
1604    BEGIN
1605        ok := pachet_robertto.gaseste_client_pret_preferential(v_id_client_citit);
1606        IF ok THEN
1607            DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul dat este eligibil pentru un pret promotional la momentul actual!');
1608        ELSE
1609            DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul dat NU este eligibil pentru un pret promotional la momentul actual!');
1610        END IF;
1611    END;
1612
1613    DECLARE
1614        v pachet_robertto.t_comenzi;
1615        BEGIN
1616            v := pachet_robertto.afis_comenzi_pref_zi(sysdate);
1617        END;
1618    /
1619
1620    DECLARE
1621        v pachet_robertto.t_comenzi;
1622        BEGIN
1623            SELECT c.id_comanda, c.pret, c.id_client BULK COLLECT INTO v
1624            FROM comanda c;
1625
1626            pachet_robertto.sortare(v);
1627        END;
1628    /
1629
1630    BEGIN
1631        pachet_robertto.f_pachet_robertto();
1632    END;
1633    /
```

The output window shows the results of the DBMS\_OUTPUT.PUT\_LINE statements and the final output of the procedure:

```
Comanda 2 a clientului cu id 3 are pretul 130
Comanda 4 a clientului cu id 4 are pretul 230
2 130
4 230
Comanda 2 are urmatoarele produse:
1 * Spaghete
1 * Pizza
1 * Ciorba de post
.....
Comanda 4 are urmatoarele produse:
2 * Spaghete
1 * Paste bolognese
1 * Ciorba de post
.....
```

Below the code and output windows, the script output and query result tabs are visible.