

# GustulLaUșă

~baza de date~

MUŞAT ANDRA

Grupa 1065



---

# CUPRINS

DESCRIEREA BAZEI DE DATE.....3
SCHEMA CONCEPTUALĂ A BAZEI DE DATE.....5
DEFINIREA SCHEMEI BAZEI DE DATE(COMENZI DDL).....6
COMENZI DML.....14
STRUCTURI DE CONTROL ȘI GESTIONAREA CURSORILOR.....36
TRATAREA EXCEPTIILOR ȘI GESTIONAREA CURSORILOR.....43
PROCEDURI, FUNCȚII ȘI PACHETE.....53
DECLANȘATORI.....69

---

## OBIECTIV

Baza de date a restaurantului online "GustulLaUșă" are ca obiectiv principal să gestioneze eficient toate aspectele operaționale ale afacerii pentru a oferi o experiență plăcută clienților și a facilita buna funcționare a întregului proces. Prin integrarea detaliilor legate de clienți, adrese, produse, comenzi, rețete, ingrediente, inventar, personal, schimb și rute, sistemul vizează o coordonare și administrare transparentă și eficientă a tuturor activităților implicate în desfășurarea restaurantului online.

## ENTITĂȚI ȘI ATRIBUTE

- \* **CLIENTI\_RESTAURANT:** id\_client **PK**, prenume\_client, nume\_client, număr\_telefon
  - \* **ADRESE:** id\_adresă **PK**, adresă, oraș, cod\_poștal
- \* **PRODUSE\_RESTAURANT:** id\_produs **PK**, nume\_produs, categorie\_produs, mărime\_produs, preț\_produs, monedă
- \* **COMENZI\_RESTAURANT:** id **PK**, id\_comandă, data\_comandă, id\_client **FK**, id\_adresă **FK**
  - \* **INVENTAR:** id **PK**, id\_produs **FK**, număr\_produse, id\_comandă **FK**
  - \* **INGREDIENTE:** id\_ingredient **PK**, nume\_ingredient, cantitate, unitate\_de\_măsură
- \* **REȚETE:** id **PK**, id\_rețetă **FK**, id\_ingredient **FK**, cantitate\_ingredient, unitate\_de\_măsură
- \* **PERSONAL:** id\_personal **PK**, prenume\_personal, nume\_personal, poziție, tarif\_pe\_oră, monedă, id\_superior
- \* **SCHIMBURI:** id\_schimb **PK**, zi, începutul\_programului, sfârșitul\_programului
  - \* **RUTE:** id **PK**, id\_rută, data **FK**, id\_schimb **FK**, id\_personal **FK**

---

# **RELAȚII ÎNTRE ENTITĂȚI**

## **\* CLIENTI\_RESTAURANT și COMENZI\_RESTAURANT**

Fiecare client poate plasa mai multe comenzi, dar fiecare comandă este asociată unui singur client.

## **\* ADRESE și COMENZI\_RESTAURANT**

Fiecare adresă poate fi asociată cu mai multe comenzi, dar fiecare comandă este legată de o singură adresă.

## **\* PRODUSE\_RESTAURANT și COMENZI\_RESTAURANT**

Produsele sunt asociate cu comenzile într-o relație de mulți la mulți prin intermediul tableei "INVENTAR". Mai multe produse pot fi asociate cu mai multe comenzi.

## **\* PRODUSE\_RESTAURANT și REȚETE**

Fiecare produs poate face parte din mai multe rețete, dar fiecare rețetă este asociată cu un singur produs.

## **\* INGREDIENTE și REȚETE**

Fiecare ingredient poate fi utilizat în mai multe rețete, dar fiecare rețetă include doar un ingredient.

## **\* PERSONAL și RUTE**

Fiecare membru al personalului poate fi asignat la mai multe rute, dar fiecare rută este asignată doar unui membru al personalului.

## **\* SCHIMBURI și RUTE**

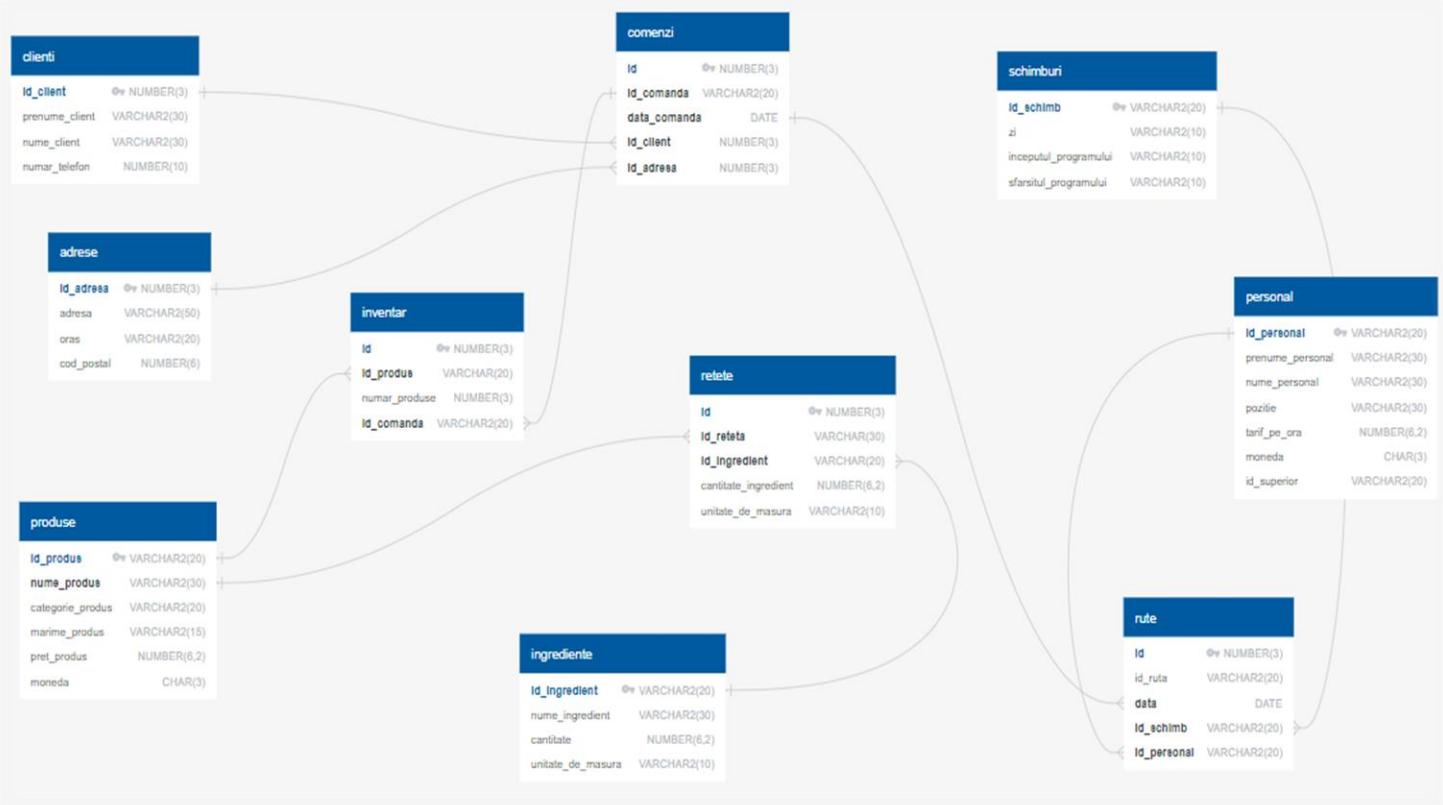
Fiecare schimb poate avea mai multe rute, dar fiecare rută este asociată doar unui schimb.

## **\* RUTE și COMENZI\_RESTAURANT**

Rutele sunt asociate cu comenzile într-o relație de mulți la mulți prin intermediul tableei "SCHIMBURI". Mai multe rute pot fi asociate cu mai multe comenzi. Data specifică a rutei este asociată cu data plasării comenzi.

Acstea relații ajută la definirea modului în care datele dintr-o tabelă sunt legate de datele dintr-o altă tabelă, furnizând o structură pentru baza de date și facilitând interogările care implică mai multe tabele.

## SCHEMA CONCEPTUALĂ A BAZEI DE DATE



---

# **DEFINIREA SCHEMEI BAZEI DE DATE (COMENZI DDL)**

## **\* CREAREA TABELEI CLIENTI\_RESTAURANT:**

```
CREATE TABLE clienti_restaurant (
    id_client NUMBER(3) CONSTRAINT clrst_id_client_pk PRIMARY KEY,
    prenume_client VARCHAR2(30) CONSTRAINT clrst_prenume_nn NOT NULL,
    nume_client VARCHAR2(30)
);
```

Table created.

## **\* CREAREA TABELEI ADRESE:**

```
CREATE TABLE adresa (
    id_adresa NUMBER(3) CONSTRAINT adrese_id_adresa_pk PRIMARY KEY,
    adresa VARCHAR2(50) CONSTRAINT adrese_adresa_nn NOT NULL,
    oras VARCHAR2(20) CONSTRAINT adrese_oras_ck CHECK (oras IN ('Bucuresti',
    'Popesti Leordeni', 'Buftea', 'Bragadiru', 'Chitila', 'Otopeni')),
    cod_postal NUMBER(6)
);
```

Table created.

## **\* CREAREA TABELEI PRODUSE\_RESTAURANT:**

```
CREATE TABLE produse_restaurant (
    id_produs VARCHAR2(20) CONSTRAINT prest_id_produs_pk PRIMARY KEY,
    nume_produs VARCHAR2(30) CONSTRAINT prest_nume_nn NOT NULL,
    categorie_produs VARCHAR2(20),
```

```
marime_produs VARCHAR2(15) CONSTRAINT prest_marime_nn NOT NULL,  
pret_produs NUMBER(6,2) CONSTRAINT prest_pret_nn NOT NULL,  
moneda CHAR(3) DEFAULT 'RON'  
);
```

Table created.

#### \* CREAREA TABELEI COMENZI\_RESTAURANT:

```
CREATE TABLE comenzi_restaurant (  
    id NUMBER(3) CONSTRAINT crest_id_comenzi_pk PRIMARY KEY,  
    id_comanda VARCHAR2(20) CONSTRAINT crest_id_comanda_nn NOT NULL,  
    data_comanda DATE CONSTRAINT crst_data_nn NOT NULL,  
    id_client NUMBER(3),  
    id_adresa NUMBER(3),  
    CONSTRAINT crest_id_client_fk FOREIGN KEY (id_client)  
        REFERENCES clienti_restaurant (id_client) ON DELETE CASCADE,  
    CONSTRAINT crest_id_adresa_fk FOREIGN KEY (id_adresa)  
        REFERENCES adrese (id_adresa) ON DELETE CASCADE  
);
```

Table created.

#### \* CREAREA TABELEI INVENTAR:

```
CREATE TABLE inventar (  
    id NUMBER(3) CONSTRAINT inv_id_inventar_PK PRIMARY KEY,  
    id_produs VARCHAR(20),  
    numar_produse NUMBER(3),  
    id_comanda VARCHAR2(20),  
    CONSTRAINT inv_id_produs_fk FOREIGN KEY (id_produs)  
        REFERENCES produse_restaurant (id_produs) ON DELETE CASCADE,  
    CONSTRAINT inv_id_comanda_fk FOREIGN KEY (id_comanda)  
        REFERENCES comenzi_restaurant (id_comanda) ON DELETE CASCADE
```

Table created.

);

\* **CREAREA TABELEI INGREDIENTE:**

```
CREATE TABLE ingrediente (
```

```
    id_ingredient VARCHAR2(20) CONSTRAINT ing_ingredient_pk PRIMARY KEY,  
    nume_ingredient VARCHAR2(30) CONSTRAINT ing_nume_nn NOT NULL,  
    cantitate NUMBER(6,2),  
    unitate_de_masura VARCHAR2(10)
```

);

Table created.

\* **CREAREA TABELEI RETETE:**

```
CREATE TABLE retete (
```

```
    id NUMBER(3) CONSTRAINT reteta_id_reteta_pk PRIMARY KEY,  
    id_reteta VARCHAR2(30) NOT NULL,  
    id_ingredient VARCHAR2(20) CONSTRAINT reteta_id_ingredient_uk UNIQUE,  
    cantitate_ingredient NUMBER(6,2),  
    unitate_de_masura VARCHAR2(10),  
    CONSTRAINT reteta_id_reteta_fk FOREIGN KEY (id_reteta)  
        REFERENCES produse_restaurant (nume_produs) ON DELETE CASCADE,  
    CONSTRAINT reteta_id_ingredient_fk FOREIGN KEY (id_ingredient)  
        REFERENCES ingrediente (id_ingredient) ON DELETE CASCADE
```

);

Table created.

\* **CREAREA TABELEI PERSONAL:**

```
CREATE TABLE personal (
```

```
    id_personal VARCHAR2(20) CONSTRAINT personal_id_personal_pk PRIMARY  
    KEY,
```

```
prenume_personal VARCHAR2(30) CONSTRAINT personal_prenume_nn NOT NULL,  
nume_personal VARCHAR2(30) CONSTRAINT personal_nume_nn NOT NULL,  
pozitie VARCHAR2(30) CONSTRAINT personal_pozitie_nn NOT NULL,  
tarif_pe_ora NUMBER(6,2) CONSTRAINT personal_tarif_nn NOT NULL,  
moneda CHAR(3) DEFAULT 'RON',  
id_superior VARCHAR2(20)  
);
```

Table created.

#### \* CREAREA TABELEI SCHIMBURI:

```
CREATE TABLE schimb (  
    id_schimb VARCHAR2(20) CONSTRAINT schimb_id_schimb_pk PRIMARY KEY,  
    zi VARCHAR2(10) CONSTRAINT schimb_zi_nn NOT NULL,  
    inceputul_programului VARCHAR2(10),  
    sfarsitul_programului VARCHAR2(10)  
);
```

Table created.

#### \* CREAREA TABELEI RUTE:

```
CREATE TABLE rute (  
    id NUMBER(3) CONSTRAINT rute_id_rute_pk PRIMARY KEY,  
    id_ruta VARCHAR2(20),  
    data DATE,  
    id_schimb VARCHAR2(20),  
    id_personal VARCHAR2(20)  
    CONSTRAINT rute_data_fk FOREIGN KEY (data)  
    REFERENCES comenzi_restaurant (data_comanda) ON DELETE CASCADE,  
    CONSTRAINT rute_id_schimb_fk FOREIGN KEY (id_schimb)
```

```
REFERENCES schimburi (id_schimb) ON DELETE CASCADE,  
CONSTRAINT rute_id_personal_fk FOREIGN KEY (id_personal)  
REFERENCES personal (id_personal) ON DELETE CASCADE  
);
```

Table created.

- \* Atașează comentariul „Clientii restaurantului sunt mulțumiti.” tablei clienti\_restaurant:

```
COMMENT ON TABLE clienti_restaurant  
IS 'Clientii restaurantului sunt mulțumiti.';
```

Statement processed.

```
SELECT table_name, comments  
FROM user_tab_comments;
```

Results	Explain	Describe	Saved SQL	History
CLIENTI				-

CLIENTI_RESTAURANT	Clientii restaurantului sunt mulțumiti.
--------------------	---

- \* Adaugă restricția de integritate NOT NULL pe coloana nume\_client a tablei clienti\_restaurant:

```
ALTER TABLE clienti_restaurant MODIFY(nume_client CONSTRAINT  
clrst_num_nn NOT NULL);
```

Table altered.

- \* Adaugă coloana varsta de tip NUMBER(1) în tabela clienti\_restaurant:

```
ALTER TABLE clienti_restaurant ADD (varsta NUMBER(1));
```

Table altered.

- \* Modifică tipul coloanei varsta a tabeliei clienti\_restaurant în NUMBER(2):

```
ALTER TABLE clienti_restaurant MODIFY(varsta NUMBER(2));
```

Table altered.

- \* Adaugă restricția de integritate NOT NULL pe coloana varsta a tabeliei clienti\_restaurant:

```
ALTER TABLE clienti_restaurant MODIFY(varsta CONSTRAINT clrst_varsta_nn  
NOT NULL);
```

Table altered.

- \* Sterge coloana varsta, inclusiv cu restricția sa, din tabela clienti\_restaurant:

```
ALTER TABLE clienti_restaurant DROP COLUMN varsta CASCADE  
CONSTRAINTS;
```

Table altered.

- \* Să se adauge coloana numar\_telefon de tip NUMBER(10) în tabela clienti\_restaurant:

```
ALTER TABLE clienti_restaurant ADD (numar_telefon  
NUMBER(10));
```

Table altered.

- \* Adaugă restricția de integritate NOT NULL pe coloana numar\_telefon a tabelei clienti\_restaurant:

```
ALTER TABLE clienti_restaurant MODIFY(numar_telefon CONSTRAINT  
clrst_nr_telefon_nn NOT NULL);
```

Table altered.

- \* Sterge tabela clienti\_restaurant, inclusiv cu restricțiile sale, apoi recuperează-o:

```
DROP TABLE clienti_restaurant CASCADE CONSTRAINTS;
```

Table dropped.

```
FLASHBACK TABLE clienti_restaurant TO BEFORE DROP;
```

- \* Să se dezactiveze restricția de integritate de pe coloana oras din tabela adrese:

```
ALTER TABLE adrese DISABLE CONSTRAINT adrese_oras_ck;
```

Table altered.

- \* Să se activeze restricția de integritate de pe coloana oras din tabela adrese:

```
ALTER TABLE adrese ENABLE CONSTRAINT adrese_oras_ck;
```

Table altered.

- \* Să se adauge restricția de integritate UNIQUE pe coloana nume\_produs a tabelei produse\_restaurant:

ALTER TABLE produse\_restaurant ADD CONSTRAINT prst\_nume\_uk UNIQUE (nume\_produs);

Table altered.

- \* Să se șteargă restricția de integritate de pe coloana data\_coamanda a tabelei comenzi\_restaurant:

ALTER TABLE comenzi\_restaurant DROP CONSTRAINT crst\_data\_nn;

Table altered.

- \* Să se adauge restricția de integritate UNIQUE pe coloana data\_coamanda a tabelei comenzi\_restaurant:

ALTER TABLE comenzi\_restaurant ADD CONSTRAINT comenzi\_data\_uk UNIQUE (data\_comanda);

Table altered.

- \* Să se adauge restricția de integritate UNIQUE pe coloana id\_coamanda a tabelei comenzi\_restaurant:

ALTER TABLE comenzi\_restaurant ADD CONSTRAINT id\_comanda\_uk UNIQUE(id\_comanda);

Table altered.

# COMENZI DML

## \* INSERAREA DATELOR ÎN TABELA CLIENTI\_RESTAURANT:

```
INSERT INTO clienti_restaurant VALUES(1, 'Ana', 'Popescu', 1234567890);
INSERT INTO clienti_restaurant VALUES(2, 'Mihai', 'Ionescu', 9876543210);
INSERT INTO clienti_restaurant VALUES(3, 'Elena', 'Radu', 5551234567);
INSERT INTO clienti_restaurant VALUES(4, 'Adrian', 'Dumitrescu', 1112223333);
INSERT INTO clienti_restaurant VALUES(5, 'Laura', 'Constantin', 4445556666);
INSERT INTO clienti_restaurant VALUES(6, 'Ionut', 'Gheorghe', 9998887777);
INSERT INTO clienti_restaurant VALUES(7, 'Andreea', 'Stancu', 6665554444);
INSERT INTO clienti_restaurant VALUES(8, 'Cristian', 'Munteanu', 3334445555);
INSERT INTO clienti_restaurant VALUES(9, 'Roxana', 'Dinu', 2223334444);
INSERT INTO clienti_restaurant VALUES(10, 'Alexandru', 'Iancu', 7778889999);
INSERT INTO clienti_restaurant VALUES(11, 'Diana', 'Dragomir', 8889990000);
INSERT INTO clienti_restaurant VALUES(12, 'Sorin', 'Avram', 1239874560);
INSERT INTO clienti_restaurant VALUES(13, 'Georgiana', 'Pavel', 4567890123);
INSERT INTO clienti_restaurant VALUES(14, 'Andrei', 'Istrate', 7890123456);
INSERT INTO clienti_restaurant VALUES(15, 'Simona', 'Nicolae', 2345678901);
```

```
17  SELECT * FROM clienti_restaurant
18  ORDER BY id_client;
```

Results			
	Explain	Describe	Saved SQL
ID_CLIENT	PRENUME_CLIENT	NUME_CLIENT	NUMAR_TELEFON
1	Ana	Popescu	1234567890
2	Mihai	Ionescu	9876543210
3	Elena	Radu	5551234567

## \* INSERAREA DATELOR ÎN TABELA ADRESE:

```
INSERT INTO adrese VALUES(10, 'Strada Victoriei, Nr. 1', 'Bucuresti', 123456);
INSERT INTO adrese VALUES(11, 'Bulevardul Unirii, Nr. 20', 'Bucuresti', 654321);
```

INSERT INTO adrese VALUES(12, 'Aleea Florilor, Nr. 15', 'Popesti Leordeni', 987654);  
 INSERT INTO adrese VALUES(13, 'Bulevardul Republicii, Nr. 7', 'Buftea', 111222);  
 INSERT INTO adrese VALUES(14, 'Strada Libertatii, Nr. 10', 'Bragadiru', 333444);  
 INSERT INTO adrese VALUES(15, 'Bulevardul Mihai Eminescu, Nr. 25', 'Chitila', 555666);  
 INSERT INTO adrese VALUES(16, 'Aleea Soarelui, Nr. 8', 'Otopeni', 777888);  
 INSERT INTO adrese VALUES(17, 'Strada Independentei, Nr. 5', 'Bucuresti', 999000);  
 INSERT INTO adrese VALUES(18, 'Bulevardul Carol I, Nr. 12', 'Popesti Leordeni', 111222);  
 INSERT INTO adrese VALUES(19, 'Strada Stefan cel Mare, Nr. 30', 'Bucuresti', 333444);  
 INSERT INTO adrese VALUES(20, 'Aleea Castanilor, Nr. 18', 'Bragadiru', 555666);  
 INSERT INTO adrese VALUES(21, 'Bulevardul Basarabiei, Nr. 3', 'Chitila', 777888);  
 INSERT INTO adrese VALUES(22, 'Strada Revolutiei, Nr. 22', 'Otopeni', 999000);  
 INSERT INTO adrese VALUES(23, 'Bulevardul Timisoara, Nr. 17', 'Bucuresti', 111222);  
 INSERT INTO adrese VALUES(24, 'Aleea Crizantemelor, Nr. 14', 'Popesti Leordeni', 333444);

```

17  SELECT * FROM adrese
18  ORDER BY id_adresa;
  
```

Results	Explain	Describe	Saved SQL	History
21	Bulevardul Basarabiei, Nr. 3	Chitila	777888	
22	Strada Revolutiei, Nr. 22	Otopeni	999000	
23	Bulevardul Timisoara, Nr. 17	Bucuresti	111222	
24	Aleea Crizantemelor, Nr. 14	Popesti Leordeni	333444	

15 rows returned in 0.00 seconds    [Download](#)

---

## \* INSERAREA DATELOR ÎN TABELA PRODUSE\_RESTAURANT:

INSERT INTO produse\_restaurant (id\_produs, nume\_produs, categorie\_produs, marime\_produs, pret\_produs) VALUES('P1', 'Pizza Margherita', 'Pizza', 'medie', 15.9);  
INSERT INTO produse\_restaurant VALUES('P2', 'Paste Bolognese', 'Paste', 'portie medie', 12.50, 'RON');  
INSERT INTO produse\_restaurant VALUES('P3', 'Burger Clasic', 'Burger', 'XL', 18.75, 'RON');  
INSERT INTO produse\_restaurant VALUES('P4', 'Salata Caesar', 'Salata', 'portie medie', 8.99, 'RON');  
INSERT INTO produse\_restaurant VALUES('P5', 'Supa de Legume', 'Supa', 'portie medie', 5.50, 'RON');  
INSERT INTO produse\_restaurant VALUES('P6', 'Pui la Gratar', 'Fel Principal', 'portie mare', 22.50, 'RON');  
INSERT INTO produse\_restaurant VALUES('P7', 'Tiramisu', 'Desert', 'mediu', 7.99, 'RON');  
INSERT INTO produse\_restaurant VALUES('P8', 'Ciorba de Burta', 'Ciorba', 'portie medie', 10.25, 'RON');  
INSERT INTO produse\_restaurant VALUES('P9', 'Sushi Combo', 'Sushi', 'portie mare', 29.99, 'RON');  
INSERT INTO produse\_restaurant VALUES('P10', 'Pasta Carbonara', 'Paste', 'portie mare', 14.75, 'RON');  
INSERT INTO produse\_restaurant VALUES('P11', 'Cheesecake', 'Desert', 'mediu', 9.25, 'RON');  
INSERT INTO produse\_restaurant VALUES('P12', 'Fish and Chips', 'Fel Principal', 'portie mare', 17.50, 'RON');  
INSERT INTO produse\_restaurant VALUES('P13', 'Sandwich Veggie', 'Sandwich', 'mediu', 8.50, 'RON');  
INSERT INTO produse\_restaurant VALUES('P14', 'Lasagna', 'Fel Principal', 'portie mare', 16.50, 'RON');  
INSERT INTO produse\_restaurant VALUES('P15', 'Sorbet de Fructe', 'Desert', 'mediu', 6.99, 'RON');  
INSERT INTO produse\_restaurant VALUES('B1', 'Apa Minerală', 'Băutură', '0.5L', 3.50, 'RON');  
INSERT INTO produse\_restaurant VALUES('B2', 'Coca-Cola', 'Suc', '0.33L', 5.00, 'RON');

```

INSERT INTO produse_restaurant VALUES('B3', 'Cafea Americano', 'Cafea', 'M', 7.99, 'RON');
INSERT INTO produse_restaurant VALUES('B4', 'Vin Rosu Sec', 'Vin', '1L', 25.99, 'RON');
INSERT INTO produse_restaurant VALUES('B5', 'Whisky Single Malt', 'Spirtoase', '50ml', 40.00, 'RON');

```

The screenshot shows the Oracle APEX interface with the following details:

- SQL Statement:**

```

22  SELECT * FROM produse_restaurant
23  ORDER BY id_produs;
24

```

- Results Grid:** Displays the following data:

P6	Pui la Gratar	Fel Principal	portie mare	22.5	RON
P7	Tiramisu	Desert	mediu	7.99	RON
P8	Ciorba de Burta	Ciorba	portie medie	10.25	RON
P9	Sushi Combo	Sushi	portie mare	29.99	RON

- Message:** 20 rows returned in 0.00 seconds
- Buttons:** Explain, Describe, Saved SQL, History, Download
- Footer:** andra\_musat62@yahoo.com, andra\_seminar\_1046\_musatandra, Copyright © 1999, 2023, Oracle and/or its affiliates., Oracle APEX 23.2

### \* INSERAREA DATELOR ÎN TABELA COMENZI\_RESTAURANT:

```

INSERT INTO comenzi_restaurant VALUES(100, 'C1001', TO_DATE('05-12-2023', 'DD-MM-YYYY'), 1, 10);
INSERT INTO comenzi_restaurant VALUES(101, 'C1002', TO_DATE('06-12-2023', 'DD-MM-YYYY'), 2, 11);
INSERT INTO comenzi_restaurant VALUES(102, 'C1003', TO_DATE('07-12-2023', 'DD-MM-YYYY'), 3, 12);
INSERT INTO comenzi_restaurant VALUES(103, 'C1004', TO_DATE('08-12-2023', 'DD-MM-YYYY'), 4, 13);
INSERT INTO comenzi_restaurant VALUES(104, 'C1005', TO_DATE('09-12-2023', 'DD-MM-YYYY'), 5, 14);
INSERT INTO comenzi_restaurant VALUES(105, 'C1006', TO_DATE('10-12-2023', 'DD-MM-YYYY'), 6, 15);
INSERT INTO comenzi_restaurant VALUES(106, 'C1007', TO_DATE('11-12-2023', 'DD-MM-YYYY'), 7, 16);

```

---

```

INSERT INTO comenzi_restaurant VALUES(107, 'C1008', TO_DATE('12-12-2023',
'DD-MM-YYYY'), 8, 17);
INSERT INTO comenzi_restaurant VALUES(108, 'C1009', TO_DATE('13-12-2023',
'DD-MM-YYYY'), 9, 18);
INSERT INTO comenzi_restaurant VALUES(109, 'C1010', TO_DATE('14-12-2023',
'DD-MM-YYYY'), 10, 19);
INSERT INTO comenzi_restaurant VALUES(110, 'C1011', TO_DATE('15-12-2023',
'DD-MM-YYYY'), 11, 20);
INSERT INTO comenzi_restaurant VALUES(111, 'C1012', TO_DATE('16-12-2023',
'DD-MM-YYYY'), 12, 21);
INSERT INTO comenzi_restaurant VALUES(112, 'C1013', TO_DATE('17-12-2023',
'DD-MM-YYYY'), 13, 22);
INSERT INTO comenzi_restaurant VALUES(113, 'C1014', TO_DATE('18-12-2023',
'DD-MM-YYYY'), 14, 23);
INSERT INTO comenzi_restaurant VALUES(114, 'C1015', TO_DATE('19-12-2023',
'DD-MM-YYYY'), 15, 24);

```

ID	ID_COMANDA	DATA_COMANDA	ID_CLIENT	ID_ADRESA
1	100 C1001	05-DEC-23	1	10
2	101 C1002	06-DEC-23	2	11
3	102 C1003	07-DEC-23	3	12
4	103 C1004	08-DEC-23	4	13
5	104 C1005	09-DEC-23	5	14
6	105 C1006	10-DEC-23	6	15
7	106 C1007	11-DEC-23	7	16
8	107 C1008	12-DEC-23	8	17
9	108 C1009	13-DEC-23	9	18

#### \* INSERAREA DATELOR ÎN TABELA INGREDIENTE:

```

INSERT INTO ingrediente VALUES ('I1', 'Aluat Pizza', 300, 'kg');
INSERT INTO ingrediente VALUES ('I2', 'Sos Tomat', 50, 'kg');
INSERT INTO ingrediente VALUES ('I3', 'Mozzarella', 100, 'kg');
INSERT INTO ingrediente VALUES ('I4', 'Rosii Cherry', 100, 'kg');

```

---

```
INSERT INTO ingrediente VALUES ('I5', 'Busuioc', 5, 'kg');
INSERT INTO ingrediente VALUES ('I6', 'Ulei de masline', 40, 'l');
INSERT INTO ingrediente VALUES ('I7', 'Paste Fainoasse', 200, 'kg');
INSERT INTO ingrediente VALUES ('I8', 'Sos Bolognese', 180, 'kg');
INSERT INTO ingrediente VALUES ('I9', 'Parmezan', 20, 'kg');
INSERT INTO ingrediente VALUES ('I10', 'Chifla Burger', 100, 'buc');
INSERT INTO ingrediente VALUES ('I11', 'Carne de Vită', 150, 'kg');
INSERT INTO ingrediente VALUES ('I12', 'Roșie', 100, 'buc');
INSERT INTO ingrediente VALUES ('I13', 'Castravete', 50, 'kg');
INSERT INTO ingrediente VALUES ('I14', 'Frunze de Salata Romaine', 60, 'kg');
INSERT INTO ingrediente VALUES ('I15', 'Piept de Pui la Gratar', 150, 'kg');
INSERT INTO ingrediente VALUES ('I16', 'Crutoane', 50, 'kg');
INSERT INTO ingrediente VALUES ('I17', 'Parmezan Ras', 30, 'kg');
INSERT INTO ingrediente VALUES ('I18', 'Sos Caesar', 40, 'kg');
INSERT INTO ingrediente VALUES ('I19', 'Morcov', 80, 'kg');
INSERT INTO ingrediente VALUES ('I20', 'Cartof', 100, 'kg');
INSERT INTO ingrediente VALUES ('I21', 'Ceapa', 50, 'kg');
INSERT INTO ingrediente VALUES ('I22', 'Dovlecel', 60, 'kg');
INSERT INTO ingrediente VALUES ('I23', 'Ardei', 40, 'kg');
INSERT INTO ingrediente VALUES ('I24', 'Usturoi', 10, 'kg');
INSERT INTO ingrediente VALUES ('I25', 'Supa de Legume', 300, 'l');
INSERT INTO ingrediente VALUES ('I26', 'Piept de Pui', 200, 'kg');
INSERT INTO ingrediente VALUES ('I27', 'Marinata pentru Pui', 50, 'kg');
INSERT INTO ingrediente VALUES ('I28', 'Ulei de Măslini', 20, 'l');
INSERT INTO ingrediente VALUES ('I29', 'Sare', 5, 'kg');
INSERT INTO ingrediente VALUES ('I30', 'Piper', 2, 'kg');
INSERT INTO ingrediente VALUES ('I31', 'Blat de Pandișpan', 150, 'kg');
INSERT INTO ingrediente VALUES ('I32', 'Cafea Neagra', 50, 'l');
INSERT INTO ingrediente VALUES ('I33', 'Mascarpone', 100, 'kg');
INSERT INTO ingrediente VALUES ('I34', 'Zahar', 30, 'kg');
INSERT INTO ingrediente VALUES ('I35', 'Cacao', 10, 'kg');
INSERT INTO ingrediente VALUES ('I36', 'Burta de Vita', 200, 'kg');
INSERT INTO ingrediente VALUES ('I37', 'Legume pentru Ciorbă', 150, 'kg');
INSERT INTO ingrediente VALUES ('I38', 'Fasole Boabe', 50, 'kg');
INSERT INTO ingrediente VALUES ('I39', 'Sfecla Rosie', 30, 'kg');
INSERT INTO ingrediente VALUES ('I40', 'Smantana', 50, 'l');
```

---

```
INSERT INTO ingrediente VALUES ('I41', 'Orez pentru Sushi', 100, 'kg');
INSERT INTO ingrediente VALUES ('I42', 'Nori', 100, 'folii');
INSERT INTO ingrediente VALUES ('I43', 'Somon', 50, 'kg');
INSERT INTO ingrediente VALUES ('I44', 'Avocado', 30, 'kg');
INSERT INTO ingrediente VALUES ('I45', 'Castravete', 20, 'kg');
INSERT INTO ingrediente VALUES ('I46', 'Paste Fettuccine', 200, 'kg');
INSERT INTO ingrediente VALUES ('I47', 'Bacon', 100, 'kg');
INSERT INTO ingrediente VALUES ('I48', 'Ou', 100, 'buc');
INSERT INTO ingrediente VALUES ('I49', 'Parmezan Ras', 30, 'kg');
INSERT INTO ingrediente VALUES ('I50', 'Smantana', 50, 'l');
INSERT INTO ingrediente VALUES ('I51', 'Biscuiti Graham', 100, 'kg');
INSERT INTO ingrediente VALUES ('I52', 'Unt', 50, 'kg');
INSERT INTO ingrediente VALUES ('I53', 'Branza de Vaca', 200, 'kg');
INSERT INTO ingrediente VALUES ('I54', 'Zahar', 50, 'kg');
INSERT INTO ingrediente VALUES ('I55', 'Ou', 28, 'buc');
INSERT INTO ingrediente VALUES ('I56', 'File de Peste', 150, 'kg');
INSERT INTO ingrediente VALUES ('I57', 'Cartofi Prajiti', 200, 'kg');
INSERT INTO ingrediente VALUES ('I58', 'Faina', 50, 'kg');
INSERT INTO ingrediente VALUES ('I59', 'Ou', 10, 'buc');
INSERT INTO ingrediente VALUES ('I60', 'Sare', 5, 'kg');
INSERT INTO ingrediente VALUES ('I61', 'Paine Integrala', 60, 'buc');
INSERT INTO ingrediente VALUES ('I62', 'Hummus', 30, 'kg');
INSERT INTO ingrediente VALUES ('I63', 'Avocado', 100, 'buc');
INSERT INTO ingrediente VALUES ('I64', 'Rosie', 100, 'buc');
INSERT INTO ingrediente VALUES ('I65', 'Salata Verde', 50, 'kg');
INSERT INTO ingrediente VALUES ('I66', 'Foile de Lasagna', 150, 'kg');
INSERT INTO ingrediente VALUES ('I67', 'Carne de Vita Tocata', 200, 'kg');
INSERT INTO ingrediente VALUES ('I68', 'Sos Bechamel', 100, 'kg');
INSERT INTO ingrediente VALUES ('I69', 'Branza Mozzarella', 50, 'kg');
INSERT INTO ingrediente VALUES ('I70', 'Parmezan Ras', 30, 'kg');
INSERT INTO ingrediente VALUES ('I71', 'Fructe de Padure', 100, 'kg');
INSERT INTO ingrediente VALUES ('I72', 'Zahar', 30, 'kg');
INSERT INTO ingrediente VALUES ('I73', 'Suc de Lamaie', 20, 'l');
```

The screenshot shows a database query results page. At the top, there are icons for search, refresh, and help. Below that, a code editor window contains the SQL command:

```
1  SELECT * FROM ingrediente;
```

Below the code editor is a table with four columns: ID, Nume, Quantitate, and Unitate. The data is as follows:

ID	Nume	Quantitate	Unitate
I70	Parmezan Ras	30	kg
I71	Fructe de Padure	100	kg
I72	Zahar	30	kg
I73	Suc de Lamaie	20	l

At the bottom left, it says "73 rows returned in 0.01 seconds". To the right, there is a "Download" link. At the very bottom, there are user details (andra\_musat62@yahoo.com, bd\_seminar\_1046\_musatandra), a language switcher (en), copyright information ("Copyright © 1999, 2023, Oracle and/or its affiliates."), and the text "Oracle APEX 23.2".

### \* INSERAREA DATELOR ÎN TABELA REȚETE:

```
INSERT INTO retete VALUES(1, 'Pizza Margherita', 'I1', 200, 'g');
INSERT INTO retete VALUES(2, 'Pizza Margherita', 'I2', 150, 'g');
INSERT INTO retete VALUES(3, 'Pizza Margherita', 'I3', 100, 'g');
INSERT INTO retete VALUES(4, 'Pizza Margherita', 'I4', 50, 'g');
INSERT INTO retete VALUES(5, 'Pizza Margherita', 'I5', 5, 'g');
INSERT INTO retete VALUES(6, 'Pizza Margherita', 'I6', 10, 'ml');
INSERT INTO retete VALUES(7, 'Paste Bolognese', 'I7', 200, 'g');
INSERT INTO retete VALUES(8, 'Paste Bolognese', 'I8', 180, 'g');
INSERT INTO retete VALUES(9, 'Paste Bolognese', 'I9', 20, 'g');
INSERT INTO retete VALUES(10, 'Burger Clasic', 'I10', 2, 'buc');
INSERT INTO retete VALUES(11, 'Burger Clasic', 'I11', 150, 'g');
INSERT INTO retete VALUES(12, 'Burger Clasic', 'I12', 1, 'buc');
INSERT INTO retete VALUES(13, 'Burger Clasic', 'I13', 50, 'g');
INSERT INTO retete VALUES(14, 'Salata Caesar', 'I14', 100, 'g');
INSERT INTO retete VALUES(15, 'Salata Caesar', 'I15', 150, 'g');
```

---

```
INSERT INTO retete VALUES(16, 'Salata Caesar', 'I16', 50, 'g');
INSERT INTO retete VALUES(17, 'Salata Caesar', 'I17', 30, 'g');
INSERT INTO retete VALUES(18, 'Salata Caesar', 'I18', 40, 'g');
INSERT INTO retete VALUES(19, 'Supa de Legume', 'I19', 80, 'g');
INSERT INTO retete VALUES(20, 'Supa de Legume', 'I20', 100, 'g');
INSERT INTO retete VALUES(21, 'Supa de Legume', 'I21', 50, 'g');
INSERT INTO retete VALUES(22, 'Supa de Legume', 'I22', 60, 'g');
INSERT INTO retete VALUES(23, 'Supa de Legume', 'I23', 40, 'g');
INSERT INTO retete VALUES(24, 'Supa de Legume', 'I24', 10, 'g');
INSERT INTO retete VALUES(25, 'Supa de Legume', 'I25', 300, 'ml');
INSERT INTO retete VALUES(26, 'Pui la Gratar', 'I26', 200, 'g');
INSERT INTO retete VALUES(27, 'Pui la Gratar', 'I27', 50, 'g');
INSERT INTO retete VALUES(28, 'Pui la Gratar', 'I28', 20, 'ml');
INSERT INTO retete VALUES(29, 'Pui la Gratar', 'I29', 5, 'g');
INSERT INTO retete VALUES(30, 'Pui la Gratar', 'I30', 2, 'g');
INSERT INTO retete VALUES(31, 'Tiramisu', 'I31', 150, 'g');
INSERT INTO retete VALUES(32, 'Tiramisu', 'I32', 50, 'ml');
INSERT INTO retete VALUES(33, 'Tiramisu', 'I33', 100, 'g');
INSERT INTO retete VALUES(34, 'Tiramisu', 'I34', 30, 'g');
INSERT INTO retete VALUES(35, 'Tiramisu', 'I35', 10, 'g');
INSERT INTO retete VALUES(36, 'Ciorba de Burta', 'I36', 200, 'g');
INSERT INTO retete VALUES(37, 'Ciorba de Burta', 'I37', 150, 'g');
INSERT INTO retete VALUES(38, 'Ciorba de Burta', 'I38', 50, 'g');
INSERT INTO retete VALUES(39, 'Ciorba de Burta', 'I39', 30, 'g');
INSERT INTO retete VALUES(40, 'Ciorba de Burta', 'I40', 50, 'ml');
INSERT INTO retete VALUES(41, 'Sushi Combo', 'I41', 100, 'g');
INSERT INTO retete VALUES(42, 'Sushi Combo', 'I42', 10, 'folii');
INSERT INTO retete VALUES(43, 'Sushi Combo', 'I43', 50, 'g');
INSERT INTO retete VALUES(44, 'Sushi Combo', 'I44', 30, 'g');
INSERT INTO retete VALUES(45, 'Sushi Combo', 'I45', 20, 'g');
INSERT INTO retete VALUES(46, 'Pasta Carbonara', 'I46', 200, 'g');
INSERT INTO retete VALUES(47, 'Pasta Carbonara', 'I47', 100, 'g');
INSERT INTO retete VALUES(48, 'Pasta Carbonara', 'I48', 2, 'buc');
INSERT INTO retete VALUES(49, 'Pasta Carbonara', 'I49', 30, 'g');
INSERT INTO retete VALUES(50, 'Pasta Carbonara', 'I50', 50, 'ml');
INSERT INTO retete VALUES(51, 'Cheesecake', 'I51', 100, 'g');
```

```

INSERT INTO retete VALUES(52, 'Cheesecake', 'I52', 50, 'g');
INSERT INTO retete VALUES(53, 'Cheesecake', 'I53', 200, 'g');
INSERT INTO retete VALUES(54, 'Cheesecake', 'I54', 50, 'g');
INSERT INTO retete VALUES(55, 'Cheesecake', 'I55', 2, 'buc');
INSERT INTO retete VALUES(56, 'Fish and Chips', 'I56', 150, 'g');
INSERT INTO retete VALUES(57, 'Fish and Chips', 'I57', 200, 'g');
INSERT INTO retete VALUES(58, 'Fish and Chips', 'I58', 50, 'g');
INSERT INTO retete VALUES(59, 'Fish and Chips', 'I59', 1, 'buc');
INSERT INTO retete VALUES(60, 'Fish and Chips', 'I60', 5, 'g');
INSERT INTO retete VALUES(61, 'Sandwich Veggie', 'I61', 2, 'felii');
INSERT INTO retete VALUES(62, 'Sandwich Veggie', 'I62', 30, 'g');
INSERT INTO retete VALUES(63, 'Sandwich Veggie', 'I63', 1, 'buc');
INSERT INTO retete VALUES(64, 'Sandwich Veggie', 'I64', 1, 'buc');
INSERT INTO retete VALUES(65, 'Sandwich Veggie', 'I65', 50, 'g');
INSERT INTO retete VALUES(66, 'Lasagna', 'I66', 150, 'g');
INSERT INTO retete VALUES(67, 'Lasagna', 'I67', 200, 'g');
INSERT INTO retete VALUES(68, 'Lasagna', 'I68', 100, 'g');
INSERT INTO retete VALUES(69, 'Lasagna', 'I69', 50, 'g');
INSERT INTO retete VALUES(70, 'Lasagna', 'I70', 30, 'g');
INSERT INTO retete VALUES(71, 'Sorbet de Fructe', 'I71', 100, 'g');
INSERT INTO retete VALUES(72, 'Sorbet de Fructe', 'I72', 30, 'g');
INSERT INTO retete VALUES(73, 'Sorbet de Fructe', 'I73', 20, 'ml');

```

Results	Explain	Describe	Saved SQL	History
ID	ID_RETETA	ID_INGREDIENT	CANTITATE_PRODUS	UNITATE_DE_MASURA
1	Pizza Margherita	I1	200	g
2	Pizza Margherita	I2	150	g
3	Pizza Margherita	I3	100	g
4	Pizza Margherita	I4	50	g

 andra\_musat62@yahoo.com  
 bd\_seminar\_1046\_musatandra
 
 Copyright © 1999, 2023, Oracle and/or its affiliates.
 Oracle APEX 23.2

---

#### \* INSERAREA DATELOR ÎN TABELA INVENTAR:

```
INSERT INTO inventar VALUES(001, 'P1', 2, 'C1001');
INSERT INTO inventar VALUES(002, 'P3', 1, 'C1002');
INSERT INTO inventar VALUES(003, 'P5', 3, 'C1003');
INSERT INTO inventar VALUES(004, 'P8', 1, 'C1004');
INSERT INTO inventar VALUES(005, 'P10', 2, 'C1005');
INSERT INTO inventar VALUES(006, 'P12', 1, 'C1006');
INSERT INTO inventar VALUES(007, 'P14', 4, 'C1007');
INSERT INTO inventar VALUES(008, 'B2', 2, 'C1008');
INSERT INTO inventar VALUES(009, 'B3', 1, 'C1009');
INSERT INTO inventar VALUES(010, 'B5', 1, 'C1010');
INSERT INTO inventar VALUES(011, 'P2', 3, 'C1011');
INSERT INTO inventar VALUES(012, 'P6', 2, 'C1012');
INSERT INTO inventar VALUES(013, 'P9', 1, 'C1013');
INSERT INTO inventar VALUES(014, 'B1', 4, 'C1014');
INSERT INTO inventar VALUES(015, 'P11', 1, 'C1015');
```

ID	ID_PRODUS	NUMAR_PRODUSE	ID_COMANDA
1	1 P1		6 C1001
2	2 P3		1 C1002
3	3 P5		3 C1003
4	4 P8		1 C1004
5	5 P10		2 C1005
6	6 P12		1 C1006
7	7 P14		4 C1007
8	8 B2		2 C1008
9	9 B3		1 C1009

---

## \* INSERAREA DATELOR ÎN TABELA PERSONAL:

INSERT INTO personal (id\_personal, prenume\_personal, nume\_personal, pozitie, tarif\_pe\_ora, id\_superior) VALUES('ANG1', 'Andrei', 'Popescu', 'Bucatar', 30, 'ANG2');  
INSERT INTO personal (id\_personal, prenume\_personal, nume\_personal, pozitie, tarif\_pe\_ora, id\_superior) VALUES('ANG2', 'Ioana', 'Radu', 'Bucatar Chef', 45.5, NULL);  
INSERT INTO personal (id\_personal, prenume\_personal, nume\_personal, pozitie, tarif\_pe\_ora, id\_superior) VALUES('ANG3', 'Mihai', 'Dumitrescu', 'Bucatar', 30, 'ANG4');  
INSERT INTO personal (id\_personal, prenume\_personal, nume\_personal, pozitie, tarif\_pe\_ora, id\_superior) VALUES('ANG4', 'Elena', 'Ionescu', 'Bucatar Chef', 45.5, NULL);  
INSERT INTO personal (id\_personal, prenume\_personal, nume\_personal, pozitie, tarif\_pe\_ora, id\_superior) VALUES('ANG5', 'Alexandru', 'Dragomir', 'Bucatar', 30, 'ANG6');  
INSERT INTO personal (id\_personal, prenume\_personal, nume\_personal, pozitie, tarif\_pe\_ora, id\_superior) VALUES('ANG6', 'Maria', 'Georgescu', 'Bucatar Chef', 45.5, NULL);  
INSERT INTO personal (id\_personal, prenume\_personal, nume\_personal, pozitie, tarif\_pe\_ora, id\_superior) VALUES('ANG7', 'Cristian', 'Stancu', 'Bucatar', 30, 'ANG8');  
INSERT INTO personal (id\_personal, prenume\_personal, nume\_personal, pozitie, tarif\_pe\_ora, id\_superior) VALUES('ANG8', 'Ana', 'Nicolescu', 'Bucatar Chef', 45.5, NULL);  
INSERT INTO personal (id\_personal, prenume\_personal, nume\_personal, pozitie, tarif\_pe\_ora, id\_superior) VALUES('ANG9', 'Dorin', 'Constantin', 'Livrator', 15, NULL);  
INSERT INTO personal (id\_personal, prenume\_personal, nume\_personal, pozitie, tarif\_pe\_ora, id\_superior) VALUES('ANG10', 'Raluca', 'Munteanu', 'Livrator', 15, NULL);  
INSERT INTO personal (id\_personal, prenume\_personal, nume\_personal, pozitie, tarif\_pe\_ora, id\_superior) VALUES('ANG11', 'George', 'Iliescu', 'Livrator', 15, NULL);  
INSERT INTO personal (id\_personal, prenume\_personal, nume\_personal, pozitie, tarif\_pe\_ora, id\_superior) VALUES('ANG12', 'Gabriela', 'Gheorghiu', 'Livrator', 15, NULL);

```

INSERT INTO personal (id_personal, prenume_personal, nume_personal, pozitie,
tarif_pe_ora, id_superior) VALUES('ANG13', 'Adrian', 'Moldovan', 'Livrator', 15,
NULL);
INSERT INTO personal (id_personal, prenume_personal, nume_personal, pozitie,
tarif_pe_ora, id_superior) VALUES('ANG14', 'Simona', 'Iancu', 'Livrator', 15, NULL);
INSERT INTO personal (id_personal, prenume_personal, nume_personal, pozitie,
tarif_pe_ora, id_superior) VALUES('ANG15', 'Florin', 'Popa', 'Livrator', 15, NULL);
INSERT INTO personal (id_personal, prenume_personal, nume_personal, pozitie,
tarif_pe_ora, id_superior) VALUES('ANG16', 'Florina', 'Stoica', 'Livrator', 15, NULL);

```

18     SELECT \* FROM personal;

Results	Explain	Describe	Saved SQL	History	
ANG15	Florin	Popa	Livrator	15	RON
ANG16	Florina	Popescu	Livrator	15	RON

16 rows returned in 0.01 seconds     [Download](#)

andra\_musat62@yahoo.com      Copyright © 1999, 2023, Oracle and/or its affiliates. All rights reserved.     Oracle APEX 23.2

### \* INSERAREA DATELOR ÎN TABELA SCHIMBURI:

```

INSERT INTO schimburi VALUES('S1', 'Luni', '10:30', '14:00');
INSERT INTO schimburi VALUES('S2', 'Luni', '18:30', '23:00');
INSERT INTO schimburi VALUES('S3', 'Marti', '10:30', '14:00');
INSERT INTO schimburi VALUES('S4', 'Marti', '18:30', '23:00');
INSERT INTO schimburi VALUES('S5', 'Miercuri', '10:30', '14:00');
INSERT INTO schimburi VALUES('S6', 'Miercuri', '18:30', '23:00');
INSERT INTO schimburi VALUES('S7', 'Joi', '10:30', '14:00');
INSERT INTO schimburi VALUES('S8', 'Joi', '18:30', '23:00');

```

```

INSERT INTO schimburi VALUES('S9', 'Vineri', '10:30','14:00');
INSERT INTO schimburi VALUES('S10', 'Vineri', '18:30', '23:00');
INSERT INTO schimburi VALUES('S11', 'Sambata', '10:30', '14:00');
INSERT INTO schimburi VALUES('S12', 'Sambata', '18:30', '23:00');
INSERT INTO schimburi VALUES('S13', 'Duminica', '10:30', '14:00');
INSERT INTO schimburi VALUES('S14', 'Duminica', '18:30', '23:00');

```

Results	Explain	Describe	Saved SQL	History
S11	Sambata	10:30		14:00
S12	Sambata	18:30		23:00
S13	Duminica	10:30		14:00
S14	Duminica	18:30		23:00
14 rows returned in 0.01 seconds		Download		
andra_musat62@yahoo.com		Copyright © 1999, 2023, Oracle and/or its	Oracle APEX 23.	

### \* INSERAREA DATELOR ÎN TABELA RUTE:

```

INSERT INTO rute VALUES(1,'R1', TO_DATE('05-12-2023', 'DD-MM-YYYY'), 'S3',
'ANG1');
INSERT INTO rute VALUES(2,'R1', TO_DATE('05-12-2023', 'DD-MM-YYYY'), 'S3',
'ANG2');
INSERT INTO rute VALUES(3,'R1', TO_DATE('05-12-2023', 'DD-MM-YYYY'), 'S3',
'ANG9');
INSERT INTO rute VALUES(4,'R1', TO_DATE('05-12-2023', 'DD-MM-YYYY'), 'S4',
'ANG3');
INSERT INTO rute VALUES(5,'R1', TO_DATE('05-12-2023', 'DD-MM-YYYY'), 'S4',
'ANG4');
INSERT INTO rute VALUES(6,'R1', TO_DATE('05-12-2023', 'DD-MM-YYYY'), 'S4',
'ANG10');
INSERT INTO rute VALUES(7,'R2', TO_DATE('06-12-2023', 'DD-MM-YYYY'), 'S5',
'ANG5');

```

---

```
INSERT INTO rute VALUES(8,'R2', TO_DATE('06-12-2023', 'DD-MM-YYYY'), 'S5',
'ANG6');
INSERT INTO rute VALUES(9,'R2', TO_DATE('06-12-2023', 'DD-MM-YYYY'), 'S5',
'ANG11');
INSERT INTO rute VALUES(10,'R2', TO_DATE('06-12-2023', 'DD-MM-YYYY'),
'S6', 'ANG7');
INSERT INTO rute VALUES(11,'R2', TO_DATE('06-12-2023', 'DD-MM-YYYY'),
'S6', 'ANG8');
INSERT INTO rute VALUES(12,'R2', TO_DATE('06-12-2023', 'DD-MM-YYYY'),
'S6', 'ANG12');
INSERT INTO rute VALUES(13,'R3', TO_DATE('07-12-2023', 'DD-MM-YYYY'),
'S7', 'ANG1');
INSERT INTO rute VALUES(14,'R3', TO_DATE('07-12-2023', 'DD-MM-YYYY'),
'S7', 'ANG2');
INSERT INTO rute VALUES(15,'R3', TO_DATE('07-12-2023', 'DD-MM-YYYY'),
'S7', 'ANG13');
INSERT INTO rute VALUES(16,'R3', TO_DATE('07-12-2023', 'DD-MM-YYYY'),
'S8', 'ANG3');
INSERT INTO rute VALUES(17,'R3', TO_DATE('07-12-2023', 'DD-MM-YYYY'),
'S8', 'ANG4');
INSERT INTO rute VALUES(18,'R3', TO_DATE('07-12-2023', 'DD-MM-YYYY'),
'S8', 'ANG14');
INSERT INTO rute VALUES(19,'R4', TO_DATE('08-12-2023', 'DD-MM-YYYY'),
'S9', 'ANG5');
INSERT INTO rute VALUES(20,'R4', TO_DATE('08-12-2023', 'DD-MM-YYYY'),
'S9', 'ANG6');
INSERT INTO rute VALUES(21,'R4', TO_DATE('08-12-2023', 'DD-MM-YYYY'),
'S9', 'ANG15');
INSERT INTO rute VALUES(22,'R4', TO_DATE('08-12-2023', 'DD-MM-YYYY'),
'S10', 'ANG7');
INSERT INTO rute VALUES(23,'R4', TO_DATE('08-12-2023', 'DD-MM-YYYY'),
'S10', 'ANG8');
INSERT INTO rute VALUES(24,'R4', TO_DATE('08-12-2023', 'DD-MM-YYYY'),
'S10', 'ANG15');
INSERT INTO rute VALUES(25,'R5', TO_DATE('09-12-2023', 'DD-MM-YYYY'),
'S11', 'ANG1');
```

---

```
INSERT INTO rute VALUES(26,'R5', TO_DATE('09-12-2023', 'DD-MM-YYYY'),  
'S11', 'ANG2');  
INSERT INTO rute VALUES(27,'R5', TO_DATE('09-12-2023', 'DD-MM-YYYY'),  
'S11', 'ANG16');  
INSERT INTO rute VALUES(28,'R5', TO_DATE('09-12-2023', 'DD-MM-YYYY'),  
'S12', 'ANG3');  
INSERT INTO rute VALUES(29,'R5', TO_DATE('09-12-2023', 'DD-MM-YYYY'),  
'S12', 'ANG4');  
INSERT INTO rute VALUES(30,'R5', TO_DATE('09-12-2023', 'DD-MM-YYYY'),  
'S12', 'ANG9');  
INSERT INTO rute VALUES(31,'R6', TO_DATE('10-12-2023', 'DD-MM-YYYY'),  
'S13', 'ANG5');  
INSERT INTO rute VALUES(32,'R6', TO_DATE('10-12-2023', 'DD-MM-YYYY'),  
'S13', 'ANG6');  
INSERT INTO rute VALUES(33,'R6', TO_DATE('10-12-2023', 'DD-MM-YYYY'),  
'S13', 'ANG10');  
INSERT INTO rute VALUES(34,'R6', TO_DATE('10-12-2023', 'DD-MM-YYYY'),  
'S14', 'ANG7');  
INSERT INTO rute VALUES(35,'R6', TO_DATE('10-12-2023', 'DD-MM-YYYY'),  
'S14', 'ANG8');  
INSERT INTO rute VALUES(36,'R6', TO_DATE('10-12-2023', 'DD-MM-YYYY'),  
'S14', 'ANG11');  
INSERT INTO rute VALUES(37,'R7', TO_DATE('11-12-2023', 'DD-MM-YYYY'),  
'S1', 'ANG1');  
INSERT INTO rute VALUES(38,'R7', TO_DATE('11-12-2023', 'DD-MM-YYYY'),  
'S1', 'ANG2');  
INSERT INTO rute VALUES(39,'R7', TO_DATE('11-12-2023', 'DD-MM-YYYY'),  
'S1', 'ANG12');  
INSERT INTO rute VALUES(40,'R7', TO_DATE('11-12-2023', 'DD-MM-YYYY'),  
'S2', 'ANG3');  
INSERT INTO rute VALUES(41,'R7', TO_DATE('11-12-2023', 'DD-MM-YYYY'),  
'S2', 'ANG4');  
INSERT INTO rute VALUES(42,'R7', TO_DATE('11-12-2023', 'DD-MM-YYYY'),  
'S2', 'ANG13');  
INSERT INTO rute VALUES(43,'R8', TO_DATE('12-12-2023', 'DD-MM-YYYY'),  
'S3', 'ANG5');
```

---

```
INSERT INTO rute VALUES(44,'R8', TO_DATE('12-12-2023', 'DD-MM-YYYY'),  
'S3', 'ANG6');  
INSERT INTO rute VALUES(45,'R8', TO_DATE('12-12-2023', 'DD-MM-YYYY'),  
'S3', 'ANG14');  
INSERT INTO rute VALUES(46,'R8', TO_DATE('12-12-2023', 'DD-MM-YYYY'),  
'S4', 'ANG7');  
INSERT INTO rute VALUES(47,'R8', TO_DATE('12-12-2023', 'DD-MM-YYYY'),  
'S4', 'ANG8');  
INSERT INTO rute VALUES(48,'R8', TO_DATE('12-12-2023', 'DD-MM-YYYY'),  
'S4', 'ANG15');  
INSERT INTO rute VALUES(49,'R9', TO_DATE('13-12-2023', 'DD-MM-YYYY'),  
'S5', 'ANG1');  
INSERT INTO rute VALUES(50,'R9', TO_DATE('13-12-2023', 'DD-MM-YYYY'),  
'S5', 'ANG2');  
INSERT INTO rute VALUES(51,'R9', TO_DATE('13-12-2023', 'DD-MM-YYYY'),  
'S5', 'ANG16');  
INSERT INTO rute VALUES(52,'R9', TO_DATE('13-12-2023', 'DD-MM-YYYY'),  
'S6', 'ANG3');  
INSERT INTO rute VALUES(53,'R9', TO_DATE('13-12-2023', 'DD-MM-YYYY'),  
'S6', 'ANG4');  
INSERT INTO rute VALUES(54,'R9', TO_DATE('13-12-2023', 'DD-MM-YYYY'),  
'S6', 'ANG9');  
INSERT INTO rute VALUES(55,'R10', TO_DATE('14-12-2023', 'DD-MM-YYYY'),  
'S7', 'ANG5');  
INSERT INTO rute VALUES(56,'R10', TO_DATE('14-12-2023', 'DD-MM-YYYY'),  
'S7', 'ANG6');  
INSERT INTO rute VALUES(57,'R10', TO_DATE('14-12-2023', 'DD-MM-YYYY'),  
'S7', 'ANG10');  
INSERT INTO rute VALUES(58,'R10', TO_DATE('14-12-2023', 'DD-MM-YYYY'),  
'S8', 'ANG7');  
INSERT INTO rute VALUES(59,'R10', TO_DATE('14-12-2023', 'DD-MM-YYYY'),  
'S8', 'ANG8');  
INSERT INTO rute VALUES(60,'R10', TO_DATE('14-12-2023', 'DD-MM-YYYY'),  
'S8', 'ANG11');  
INSERT INTO rute VALUES(61,'R11', TO_DATE('15-12-2023', 'DD-MM-YYYY'),  
'S9', 'ANG1');
```

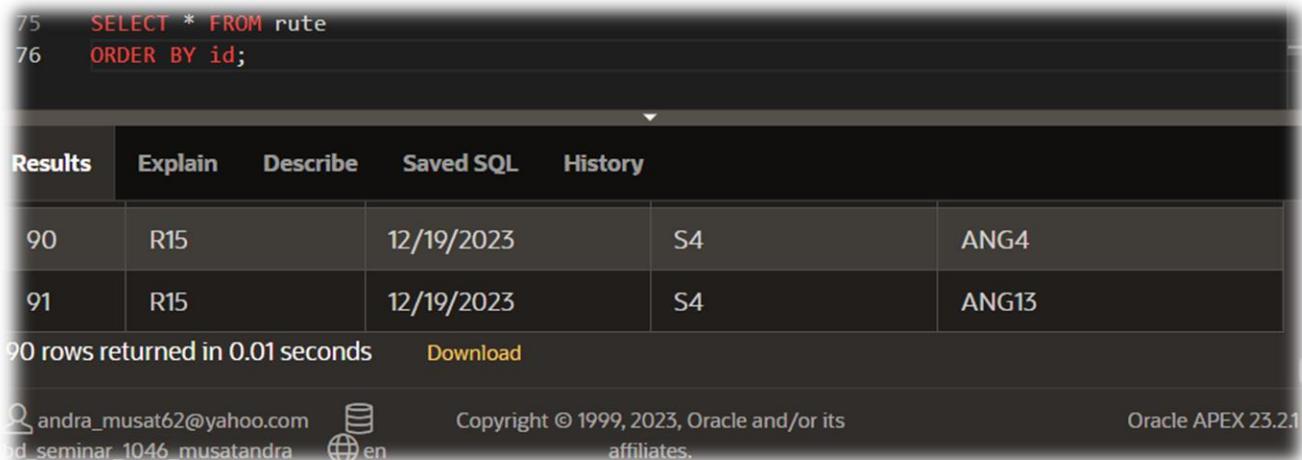
---

```
INSERT INTO rute VALUES(62,'R11', TO_DATE('15-12-2023', 'DD-MM-YYYY'),  
'S9', 'ANG2');  
INSERT INTO rute VALUES(63,'R11', TO_DATE('15-12-2023', 'DD-MM-YYYY'),  
'S9', 'ANG12');  
INSERT INTO rute VALUES(64,'R11', TO_DATE('15-12-2023', 'DD-MM-YYYY'),  
'S10', 'ANG3');  
INSERT INTO rute VALUES(65,'R11', TO_DATE('15-12-2023', 'DD-MM-YYYY'),  
'S10', 'ANG4');  
INSERT INTO rute VALUES(66,'R11', TO_DATE('15-12-2023', 'DD-MM-YYYY'),  
'S10', 'ANG13');  
INSERT INTO rute VALUES(67,'R12', TO_DATE('16-12-2023', 'DD-MM-YYYY'),  
'S11', 'ANG5');  
INSERT INTO rute VALUES(68,'R12', TO_DATE('16-12-2023', 'DD-MM-YYYY'),  
'S11', 'ANG6');  
INSERT INTO rute VALUES(69,'R12', TO_DATE('16-12-2023', 'DD-MM-YYYY'),  
'S11', 'ANG14');  
INSERT INTO rute VALUES(71,'R12', TO_DATE('16-12-2023', 'DD-MM-YYYY'),  
'S12', 'ANG7');  
INSERT INTO rute VALUES(72,'R12', TO_DATE('16-12-2023', 'DD-MM-YYYY'),  
'S12', 'ANG8');  
INSERT INTO rute VALUES(73,'R12', TO_DATE('16-12-2023', 'DD-MM-YYYY'),  
'S12', 'ANG15');  
INSERT INTO rute VALUES(74,'R13', TO_DATE('17-12-2023', 'DD-MM-YYYY'),  
'S13', 'ANG1');  
INSERT INTO rute VALUES(75,'R13', TO_DATE('17-12-2023', 'DD-MM-YYYY'),  
'S13', 'ANG2');  
INSERT INTO rute VALUES(76,'R13', TO_DATE('17-12-2023', 'DD-MM-YYYY'),  
'S13', 'ANG16');  
INSERT INTO rute VALUES(77,'R13', TO_DATE('17-12-2023', 'DD-MM-YYYY'),  
'S14', 'ANG3');  
INSERT INTO rute VALUES(78,'R13', TO_DATE('17-12-2023', 'DD-MM-YYYY'),  
'S14', 'ANG4');  
INSERT INTO rute VALUES(79,'R13', TO_DATE('17-12-2023', 'DD-MM-YYYY'),  
'S14', 'ANG9');  
INSERT INTO rute VALUES(80,'R14', TO_DATE('18-12-2023', 'DD-MM-YYYY'),  
'S1', 'ANG5');
```

```

INSERT INTO rute VALUES(81,'R14', TO_DATE('18-12-2023', 'DD-MM-YYYY'),
'S1', 'ANG6');
INSERT INTO rute VALUES(82,'R14', TO_DATE('18-12-2023', 'DD-MM-YYYY'),
'S1', 'ANG10');
INSERT INTO rute VALUES(83,'R14', TO_DATE('18-12-2023', 'DD-MM-YYYY'),
'S2', 'ANG7');
INSERT INTO rute VALUES(84,'R14', TO_DATE('18-12-2023', 'DD-MM-YYYY'),
'S2', 'ANG8');
INSERT INTO rute VALUES(85,'R14', TO_DATE('18-12-2023', 'DD-MM-YYYY'),
'S2', 'ANG11');
INSERT INTO rute VALUES(86,'R15', TO_DATE('19-12-2023', 'DD-MM-YYYY'),
'S3', 'ANG1');
INSERT INTO rute VALUES(87,'R15', TO_DATE('19-12-2023', 'DD-MM-YYYY'),
'S3', 'ANG2');
INSERT INTO rute VALUES(88,'R15', TO_DATE('19-12-2023', 'DD-MM-YYYY'),
'S3', 'ANG12');
INSERT INTO rute VALUES(89,'R15', TO_DATE('19-12-2023', 'DD-MM-YYYY'),
'S4', 'ANG3');
INSERT INTO rute VALUES(90,'R15', TO_DATE('19-12-2023', 'DD-MM-YYYY'),
'S4', 'ANG4');
INSERT INTO rute VALUES(91,'R15', TO_DATE('19-12-2023', 'DD-MM-YYYY'),
'S4', 'ANG13');

```



The screenshot shows an Oracle APEX application interface. At the top, there is a code editor window containing two SQL statements:

```

75  SELECT * FROM rute
76  ORDER BY id;

```

Below the code editor is a results grid. The grid has a header row with tabs: **Results**, Explain, Describe, Saved SQL, and History. The **Results** tab is selected. The data grid contains two rows of results:

90	R15	12/19/2023	S4	ANG4
91	R15	12/19/2023	S4	ANG13

At the bottom of the results grid, it says "90 rows returned in 0.01 seconds" and has a "Download" link.

At the very bottom of the page, there is footer information:

- User icon and email: andra\_musat62@yahoo.com
- User icon and name: and\_seminar\_1046\_musatandra
- Page language: en
- Copyright notice: Copyright © 1999, 2023, Oracle and/or its affiliates.
- Page version: Oracle APEX 23.2.1

\* Modifică numele ingredientului "Crutoane" la "Crutoane Crocante":

```
UPDATE ingrediente  
SET nume_ingredient = 'Crutoane Crocante'  
WHERE id_ingredient = 'I16';
```

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

```
1 UPDATE Ingrediente  
2 SET nume_ingredient = 'Crutoane Crocante'  
3 WHERE id_ingredient = 'I16';  
4
```

Below the code, there is a results panel with the following output:

- Results tab is selected.
- 1 row(s) updated.
- 0.02 seconds
- Andra\_musat62@yahoo.com
- Copyright © 1999, 2023, Oracle and/or its affiliates.

\* Modifică cantitatea pentru "Mozzarella" la 120 kg:

```
UPDATE ingrediente  
SET cantitate = 120  
WHERE nume_ingredient = 'Mozzarella';
```

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

```
1 UPDATE Ingrediente  
2 SET cantitate = 120  
3 WHERE nume_ingredient = 'Mozzarella';  
4  
5
```

Below the code, there is a results panel with the following output:

- Results tab is selected.
- 1 row(s) updated.
- 0.01 seconds
- Andra\_musat62@yahoo.com
- Copyright © 1999, 2023, Oracle and/or its affiliates.

- \* Modifică cantitatea pentru toate ingredientele de tip 'kg' pentru a le crește cu 10%:

UPDATE ingrediente

SET cantitate = cantitate \* 1.1

WHERE unitate\_de\_masura = 'kg';

```

1 UPDATE ingrediente
2 SET cantitate = cantitate * 1.1
3 WHERE unitate_de_masura = 'kg';
4

```

Results Explain Describe Saved SQL History

57 row(s) updated.

0.01 seconds

andra\_musat62@yahoo.com and\_seminar\_1046\_musatandra en Copyright © 1999, 2023, Oracle and its affiliates.

- \* Modifică tariful pe oră pentru toți angajații cu poziția care conține cuvântul 'Chef', astfel încât tariful lor să crească cu 10%.

UPDATE personal

SET tarif\_pe\_ora = tarif\_pe\_ora \* 1.1

WHERE UPPER(pozitie) LIKE '%CHEF%';

```

7 UPDATE personal
8 SET tarif_pe_ora = tarif_pe_ora * 1.1
9 WHERE UPPER(pozitie) LIKE '%CHEF%';
10

```

Results Explain Describe Saved SQL History

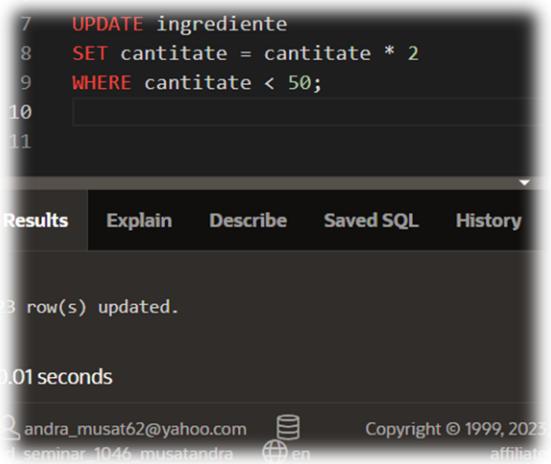
4 row(s) updated.

0.02 seconds

andra\_musat62@yahoo.com and\_seminar\_1046\_musatandra en Copyright © 1999, 2023, Oracle and its affiliates.

- \* Modifică cantitatea pentru toate ingredientele cu o cantitate sub 50 kg, dublându-le:

UPDATE ingrediente  
SET cantitate = cantitate \* 2  
WHERE cantitate < 50;



```

7   UPDATE ingrediente
8     SET cantitate = cantitate * 2
9    WHERE cantitate < 50;
10
11

```

Results Explain Describe Saved SQL History

3 row(s) updated.

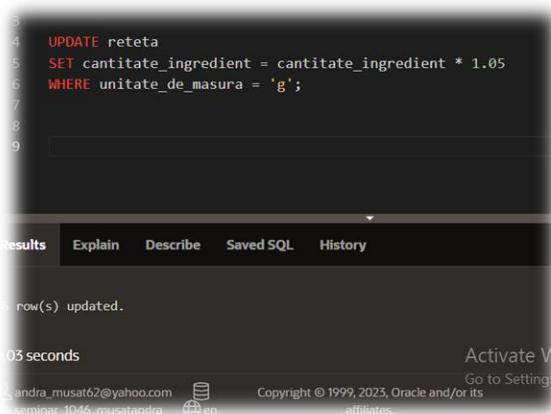
0.01 seconds

andra\_musat62@yahoo.com Copyright © 1999, 2023

1\_seminar\_1046\_musatandra en

- \* Modifică cantitatea pentru toate ingredientele de tip 'g' pentru a le crește cu 5%:

UPDATE retete  
SET cantitate\_ingredient = cantitate\_ingredient  
\* 1.05  
WHERE unitate\_de\_masura = 'g';



```

8
9   UPDATE reteta
10    SET cantitate_ingredient = cantitate_ingredient * 1.05
11   WHERE unitate_de_masura = 'g';
12
13

```

Results Explain Describe Saved SQL History

0 row(s) updated.

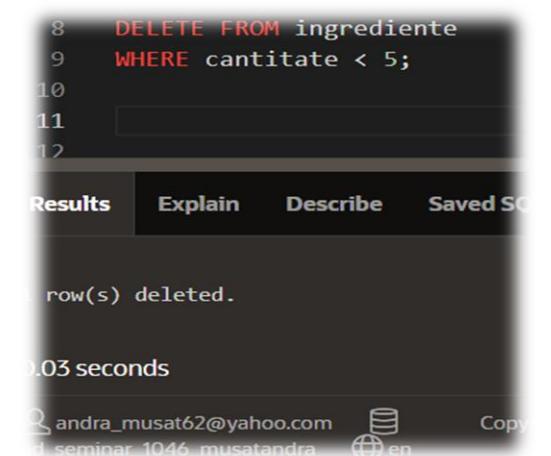
0.03 seconds

andra\_musat62@yahoo.com Copyright © 1999, 2023, Oracle and/or its

1\_seminar\_1046\_musatandra en affiliates.

- \* Șterge toate înregistrările pentru ingredientele cu o cantitate sub 5 kg:

DELETE FROM ingrediente  
WHERE cantitate < 5;



```

8   DELETE FROM ingrediente
9    WHERE cantitate < 5;
10
11
12

```

Results Explain Describe Saved SQL History

0 row(s) deleted.

0.03 seconds

andra\_musat62@yahoo.com Copyright © 1999, 2023, Oracle and/or its

1\_seminar\_1046\_musatandra en affiliates.

---

# STRUCTURI DE CONTROL ȘI GESTIONAREA CURSORILOR

- \* Scrie un bloc PL/SQL care să permită utilizatorului să introducă id-ul comenzi de la tastatură și afișează detaliile comenzi respective, inclusiv data comenzi, numele complet al clientului, produsul comandat, numărul de produse comandate, adresa de livrare, orașul și codul poștal. De asemenea, adaugă o verificare pentru a afișa un mesaj dacă comanda conține mai mult de un produs comandat sau un singur produs comandat.

```
SET SERVEROUTPUT ON
```

```
ACCEPT id PROMPT 'Introduceți id-ul comenzi de la tastatură: '
```

```
DECLARE
```

```
    v_id_comanda comenzi_restaurant.id_comanda%TYPE:='&id';
```

```
    CURSOR c_comenzi IS SELECT id_comanda, data_comanda, id_client, nume_client,  
    prenume_client, nume_produs, numar_produse, adresa, oras, cod_postal
```

```
        FROM comenzi_restaurant JOIN inventar USING (id_comanda)
```

```
        JOIN clienti_restaurant USING (id_client)
```

```
        JOIN produse_restaurant USING (id_produs)
```

```
        JOIN adrese USING (id_adresa)
```

```
        WHERE id_comanda=v_id_comanda;
```

```
    v c_comenzi%ROWTYPE;
```

```
BEGIN
```

```
    OPEN c_comenzi;
```

```
LOOP
```

```
    FETCH c_comenzi INTO v;
```

```
    EXIT WHEN c_comenzi%NOTFOUND;
```

```
    DBMS_OUTPUT.put_line('Detalii comanda:');
```

```

DBMS_OUTPUT.put_line('Id-ul comenzii: ' || v_id_comanda);
DBMS_OUTPUT.put_line('Data comenzii: ' || v.data_comanda);
DBMS_OUTPUT.put_line('Numele clientului: ' || v.nume_client || ' ' || v.prenume_client);
DBMS_OUTPUT.put_line('Produsul comandat: ' || v.nume_produs);
DBMS_OUTPUT.put_line('Numarul de produse comandate: ' || v.numar_produse);
DBMS_OUTPUT.put_line('Adresa: ' || v.adresa);
DBMS_OUTPUT.put_line('Orasul: ' || v.oras);
DBMS_OUTPUT.put_line('Codul postal: ' || v.cod_postal);
IF v.numar_produse>1 THEN
DBMS_OUTPUT.put_line('Aceasta comanda contine mai multe produse comandate.');
ELSE
DBMS_OUTPUT.put_line('Aceasta comanda contine un produs comandat.');
END IF;
END LOOP;
CLOSE c_comenzi;
END;

```

The screenshot shows the Oracle SQL Developer interface. In the top window (Worksheet), a PL/SQL block is displayed. The code uses a cursor loop to process multiple commands from a cursor named 'c\_comenzi'. It prints various details about each command, such as ID, date, client name, product name, address, city, and zip code. It also checks if there are more than one item in the command. In the bottom window (Script Output), the results of the execution are shown, including the command details and a message indicating that the current command contains more items.

```

/
  Worksheet | Query Builder
  LOOP
    FETCH c_comenzi INTO v;
    EXIT WHEN c_comenzi%NOTFOUND;
    DBMS_OUTPUT.put_line('Detalii comanda: ');
    DBMS_OUTPUT.put_line('Id-ul comenzii: ' || v_id_comanda);
    DBMS_OUTPUT.put_line('Data comenzii: ' || v.data_comanda);
    DBMS_OUTPUT.put_line('Numele clientului: ' || v.nume_client || ' ' || v.prenume_client);
    DBMS_OUTPUT.put_line('Produsul comandat: ' || v.nume_produs);
    DBMS_OUTPUT.put_line('Numarul de produse comandate: ' || v.numar_produse);
    DBMS_OUTPUT.put_line('Adresa: ' || v.adresa);
    DBMS_OUTPUT.put_line('Orasul: ' || v.oras);
    DBMS_OUTPUT.put_line('Codul postal: ' || v.cod_postal);
    IF v.numar_produse>1 THEN
      DBMS_OUTPUT.put_line('Aceasta comanda contine mai multe produse comandate.');
    ELSE
      DBMS_OUTPUT.put_line('Aceasta comanda contine un produs comandat.');
    END IF;
  END LOOP;
  CLOSE c_comenzi;
END;

Script Output x
Task completed in 2.738 seconds
END;
Detalii comanda:
Id-ul comenzii: C1003
Data comenzii: 07-DEC-23
Numele clientului: Radu Elena
Produsul comandat: Supa de Legume
Numarul de produse comandate: 3
Adresa: Aleea Florilor, Nr. 15
Orasul: Popesti Leordeni
Codul postal: 987654
Aceasta comanda contine mai multe produse comandate.

```

\* Scrie un bloc PL/SQL în care să calculezi suma totală a tuturor comenzi.

```

SET SERVEROUTPUT ON

DECLARE
    v_suma_totala NUMBER:=0;

BEGIN
    FOR comanda IN (SELECT * FROM comenzi_restaurant) LOOP
        DECLARE
            v_pret_total NUMBER;
        BEGIN
            SELECT SUM(pret_produc * numar_produse)
            INTO v_pret_total
            FROM produse_restaurant
            JOIN inventar USING (id_produc)
            WHERE id_comanda = comanda.id_comanda;
            v_suma_totala:=v_suma_totala+v_pret_total;
        END;
    END LOOP;
    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista comenzi.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Suma totala a tuturor comenzi este: ' || v_suma_totala);
    END IF;
END;
/

```

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, there are tabs for 'Welcome Page', 'Andra', 'SQL INVENTAR', and 'Query Builder'. Below the tabs, the 'Worksheet' tab is active, displaying the PL/SQL code. The 'Script Output' tab at the bottom shows the output of the executed code, which is 'Suma totala a tuturor comenzi este: 384.03'. A status message at the bottom of the output tab says 'Task completed in 0.111 seconds'.

- \* Scrie un bloc PL/SQL prin care să afișezi informații ale produselor cu cantitatea disponibilă mai mică de 100 kg.

```

SET SERVEROUTPUT ON

DECLARE
    v_nr NUMBER:=0;
    CURSOR c_ingredient IS SELECT * FROM ingrediente;
    v_ingredient c_ingredient%ROWTYPE;

BEGIN
    OPEN c_ingredient;
    LOOP
        FETCH c_ingredient INTO v_ingredient;
        EXIT WHEN c_ingredient%NOTFOUND;
        IF v_ingredient.cantitate<100 THEN
            v_nr:=v_nr+1;
            DBMS_OUTPUT.PUT_LINE('Produs ' || v_nr || ':');
            DBMS_OUTPUT.PUT_LINE('ID: ' || v_ingredient.id_ingredient);
            DBMS_OUTPUT.PUT_LINE('Nume: ' || v_ingredient.nume_ingredient);
            DBMS_OUTPUT.PUT_LINE('Cantitate disponibila: ' || v_ingredient.cantitate || ' ' ||
                v_ingredient.unitate_de_masura);
        END IF;
    END LOOP;
    CLOSE c_ingredient;
    IF v_nr=0 THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista produse cu cantitatea disponibila mai mica de 100 kg.');
    END IF;
END;/
```

The screenshot shows the Oracle SQL Developer interface with a worksheet tab open. The code in the worksheet is:

```

SET SERVEROUTPUT ON
DECLARE
    v_nr NUMBER:=0;
    CURSOR c_ingredient IS SELECT * FROM ingrediente;
    v_ingredient c_ingredient%ROWTYPE;
BEGIN
    OPEN c_ingredient;

```

The script output window shows the results of the query:

```

Task completed in 0.154 seconds
cantitate disponibila: 60 buc
Produs 37:
ID: 162
Nume: Hummus
cantitate disponibila: 30 kg
Produs 38:
ID: 165
Nume: Salata Verde
cantitate disponibila: 50 kg
Produs 39:
ID: 169
Nume: Branza Mozzarella
cantitate disponibila: 50 kg
Produs 40:
ID: 170
Nume: Parmezan Ras
cantitate disponibila: 30 kg
Produs 41:
ID: 172
Nume: Zahar
cantitate disponibila: 30 kg
Produs 42:
ID: 173

```

- \* Scrie un bloc PL/SQL prin care să actualizezi cantitatea unui ingredient specificat prin id.

SET SERVEROUTPUT ON

ACCEPT id PROMPT 'Introduceti id-ul ingredientului: '

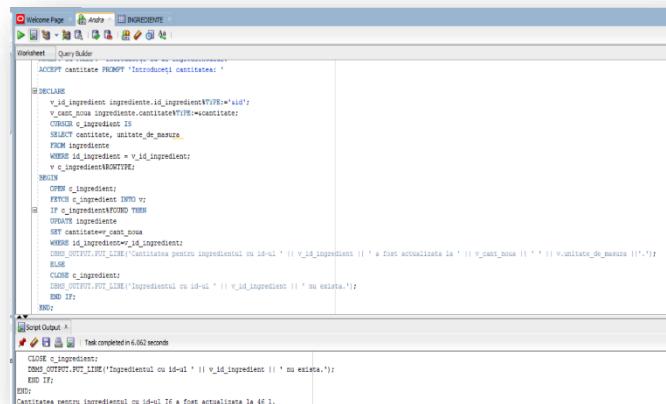
ACCEPT cantitate PROMPT 'Introduceti cantitatea: '

**DECLARE**

```

v_id_ingredient ingredient.id_ingredient%TYPE:='&id';
v_cant_noua ingredient.cantitate%TYPE:='&cantitate;
CURSOR c_ingredient IS
SELECT cantitate, unitate_de_masura
FROM ingrediente
WHERE id_ingredient = v_id_ingredient;
v c_ingredient%ROWTYPE;
BEGIN
OPEN c_ingredient;
FETCH c_ingredient INTO v;
IF c_ingredient%FOUND THEN
UPDATE ingrediente
SET cantitate=v_cant_noua
WHERE id_ingredient=v_id_ingredient;
DBMS_OUTPUT.PUT_LINE('Cantitatea pentru ingredie-
actualizata la ' || v_cant_noua || '' ||
v.unitate_de_masura ||'.');
ELSE
CLOSE c_ingredient;
DBMS_OUTPUT.PUT_LINE('Ingredientul
cu id-ul ' || v_id_ingredient || ' nu exista.');
END IF;
END;
/

```



- \* Scrie un bloc PL/SQL prin care să calculezi orele lucrate de fiecare angajat în fiecare zi.

```

SET SERVEROUTPUT ON
DECLARE
    v_id_personal personal.id_personal%TYPE;
    v_ore_lucrare NUMBER:=0;
BEGIN
    FOR personal_rec IN (SELECT DISTINCT id_personal FROM route) LOOP
        v_id_personal:=personal_rec.id_personal;
        FOR ruta IN (SELECT id_schimb, EXTRACT(HOUR FROM
TO_TIMESTAMP(sfarsitul_programului, 'HH24:MI')) - EXTRACT(HOUR FROM
TO_TIMESTAMP(inceputul_programului, 'HH24:MI')) AS ore_lucrare
        FROM route
        JOIN schimburi USING (id_schimb)
        WHERE id_personal=v_id_personal) LOOP
            v_ore_lucrare:=v_ore_lucrare+ruta.ore_lucrare;
        END LOOP;
        DBMS_OUTPUT.PUT_LINE('Angajatul '|| v_id_personal || ' a lucrat '|| v_ore_lucrare || ' ore.');
    END LOOP;
END;
/

```

The screenshot shows the Oracle SQL Developer interface. In the top-left corner, there's a toolbar with various icons. Below it is a tab bar with 'Worksheet' and 'Query Builder' selected. The main area contains a code editor with the following PL/SQL script:

```

SET SERVEROUTPUT ON
DECLARE
    v_id_personal personal.id_personal%TYPE;
    v_ore_lucrare NUMBER:=0;
BEGIN
    FOR personal_rec IN (SELECT DISTINCT id_personal FROM route) LOOP
        v_id_personal:=personal_rec.id_personal;
        FOR ruta IN (SELECT id_schimb, EXTRACT(HOUR FROM TO_TIMESTAMP(sfarsitul_programului, 'HH24:MI')) - EXTRACT(HOUR FROM TO_TIMESTAMP(inceputul_programului, 'HH24:MI')) AS ore_lucrare
        FROM route
        JOIN schimburi USING (id_schimb)
        WHERE id_personal=v_id_personal) LOOP
            v_ore_lucrare:=v_ore_lucrare+ruta.ore_lucrare;
        END LOOP;
        DBMS_OUTPUT.PUT_LINE('Angajatul '|| v_id_personal || ' a lucrat '|| v_ore_lucrare || ' ore.');
    END LOOP;
END;
/

```

Below the code editor is a 'Script Output' window. It displays the results of the execution:

```

Angajatul ANG1 a lucrat 32 ore.
Angajatul ANG2 a lucrat 64 ore.
Angajatul ANG9 a lucrat 83 ore.
Angajatul ANG3 a lucrat 123 ore.
Angajatul ANG4 a lucrat 163 ore.
Angajatul ANG10 a lucrat 180 ore.
Angajatul ANG5 a lucrat 208 ore.
Angajatul ANG6 a lucrat 236 ore.
Angajatul ANG7 a lucrat 268 ore.

```

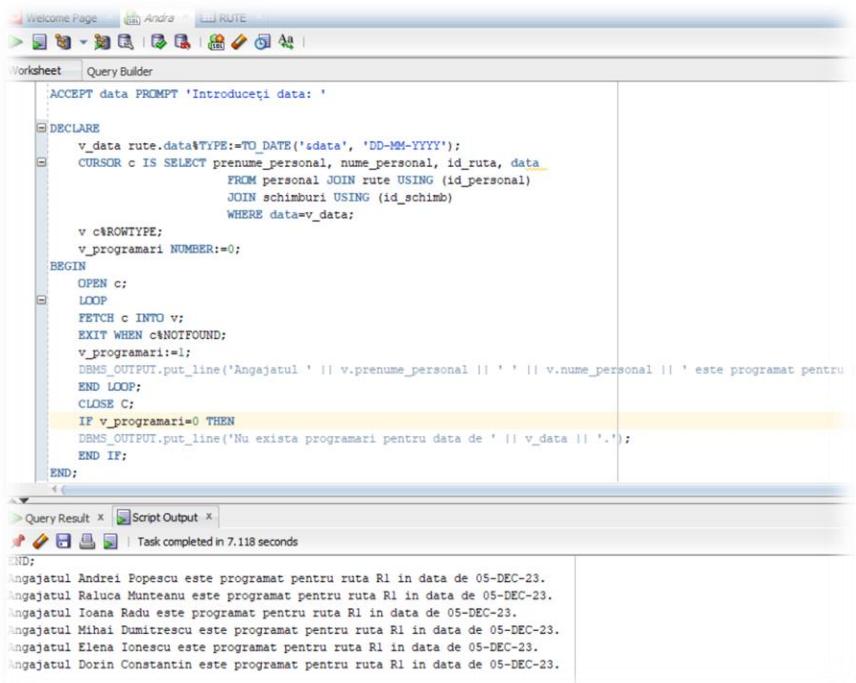
At the bottom of the 'Script Output' window, a message says 'Task completed in 0.1 seconds'.

- \* Scrie un bloc PL/SQL în care să se solicite utilizatorului introducerea unei date și să se afișeze toți angajații care sunt programăți pentru acea dată. Dacă nu există programări pentru data respectivă trebuie să se afișeze un mesaj corespunzător.

```

SET SERVEROUTPUT ON
ACCEPT data PROMPT 'Introduceți data: '
DECLARE
    v_data DATE:=TO_DATE('&data', 'DD-MM-YYYY');
    CURSOR c IS SELECT prenume_personal, nume_personal, id_ruta, data
        FROM personal JOIN ruta USING (id_personal)
        JOIN schimburi USING (id_schimb)
        WHERE data=v_data;
    v c%ROWTYPE;
    v_programari NUMBER:=0;
BEGIN
    OPEN c;
    LOOP
        FETCH c INTO v;
        EXIT WHEN c%NOTFOUND;
        v_programari:=1;
        DBMS_OUTPUT.put_line('Angajatul ' || v.prenume_personal || ' ' || v.nume_personal || ' este
programat pentru ruta ' || v.id_ruta || ' in data de ' || v_data || ' ');
    END LOOP;
    CLOSE C;
    IF v_programari=0 THEN
        DBMS_OUTPUT.put_line('Nu
exista programari pentru data de ' ||
v_data || ' ');
    END IF;
END;
/

```



---

# TRATAREA EXCEPTIILOR ȘI GESTIONAREA CURSORILOR

- \* Scrie un bloc PL/SQL care să actualizeze cantitatea tuturor ingredientelor din baza de date cu o anumită unitate de măsură dată de utilizator. Dacă unitatea de măsură introdusă nu există în baza de date, afișează un mesaj corespunzător.

```
SET SERVEROUTPUT ON
ACCEPT unitate PROMPT 'Introduceți unitatea de masura: '
DECLARE
    v_unitate ingredientе.unitate_de_masura%TYPE:='&unitate';
    unitate_inexistenta EXCEPTION;
BEGIN
    UPDATE ingredientе
    SET cantitate=cantitate*1.05
    WHERE unitate_de_masura=v_unitate;
    IF SQL%ROWCOUNT=0 THEN
        RAISE unitate_inexistenta;
    END IF;
    DBMS_OUTPUT.put_line('Actualizarea a fost efectuata.');
EXCEPTION
    WHEN unitate_inexistenta THEN
        DBMS_OUTPUT.put_line('Nu există unitatea de masură introdusa în baza de date.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.put_line('Actualizarea nu a putut fi efectuata.');
END;
/
```

→ introduc ‘ml’, unitate de măsură care nu există în baza mea de date:

```

Welcome Page Andra INGREDIENTE
Worksheet Query Builder
v_unitate ingredientе.unitate_de_masura$TYPE:='unitate';
unitate_inexistenta EXCEPTION;
BEGIN
    UPDATE ingredientе
    SET cantitate=cantitate*1.05
    WHERE unitate_de_masura=v_unitate;
    IF SQL%ROWCOUNT=0 THEN
        RAISE unitate_inexistenta;
    END IF;
    DBMS_OUTPUT.put_line('Actualizarea a fost efectuata.');
EXCEPTION
    WHEN unitate_inexistenta THEN
        DBMS_OUTPUT.put_line('Nu există unitatea de masură introdusă în');
    WHEN OTHERS THEN
        DBMS_OUTPUT.put_line('Actualizarea nu a putut fi efectuată.');
END;

```

Script Output x | Task completed in 1.438 seconds

```

IF SQL%ROWCOUNT=0 THEN
    RAISE unitate_inexistenta;
END IF;
DBMS_OUTPUT.put_line('Actualizarea a fost efectuata.');

EXCEPTION
    WHEN unitate_inexistenta THEN
        DBMS_OUTPUT.put_line('Nu există unitatea de masură introdusă în');
    WHEN OTHERS THEN
        DBMS_OUTPUT.put_line('Actualizarea nu a putut fi efectuată.');
END;
Nu există unitatea de masură introdusă în baza de date.

PL/SQL procedure successfully completed.

```

→ introduc ‘kg’, unitate de măsură care există în baza mea de date:

```

Welcome Page Andra INGREDIENTE
Worksheet Query Builder
v_unitate ingredientе.unitate_de_masura$TYPE:='unitate';
unitate_inexistenta EXCEPTION;
BEGIN
    UPDATE ingredientе
    SET cantitate=cantitate*1.05
    WHERE unitate_de_masura=v_unitate;
    IF SQL%ROWCOUNT=0 THEN
        RAISE unitate_inexistenta;
    END IF;
    DBMS_OUTPUT.put_line('Actualizarea a fost efectuata.');
EXCEPTION
    WHEN unitate_inexistenta THEN
        DBMS_OUTPUT.put_line('Nu există unitatea de masură introdusă în');
    WHEN OTHERS THEN
        DBMS_OUTPUT.put_line('Actualizarea nu a putut fi efectuată.');
END;

```

Script Output x | Task completed in 4.358 seconds

```

IF SQL%ROWCOUNT=0 THEN
    RAISE unitate_inexistenta;
END IF;
DBMS_OUTPUT.put_line('Actualizarea a fost efectuata.');

EXCEPTION
    WHEN unitate_inexistenta THEN
        DBMS_OUTPUT.put_line('Nu există unitatea de masură introdusă în');
    WHEN OTHERS THEN
        DBMS_OUTPUT.put_line('Actualizarea nu a putut fi efectuată.');
END;
Actualizarea a fost efectuata.

PL/SQL procedure successfully completed.

```

- 
- \* Scrie un bloc PL/SQL care să permită actualizarea numărului de produse comandate pentru un anumit produs din inventar. Utilizatorul va introduce id-ul produsului și programul va verifica dacă există acest id în baza de date. Dacă există, va actualiza numărul de produse la 6 dacă acesta este mai mic de 5. În caz contrar, va afișa un mesaj corespunzător.

```
SET SERVEROUTPUT ON
ACCEPT id PROMPT 'Introduceți id-ul produsului: '
DECLARE
    v_id produse_restaurant.id_produs%TYPE:='&id';
    CURSOR c IS SELECT id_produs, nume_produs, numar_produse
        FROM inventar JOIN produse_restaurant USING (id_produs)
        WHERE id_produs=UPPER(v_id);
    v_nume produse_restaurant.nume_produs%TYPE;
    v_numar inventar.numar_produse%TYPE;
    produs_inexistent EXCEPTION;
BEGIN
    OPEN c;
    FETCH c INTO v_id, v_nume, v_numar;
    IF c%NOTFOUND THEN
        RAISE produs_inexistent;
    END IF;
    IF v_numar<5 THEN
        UPDATE inventar
        SET numar_produse=6
        WHERE id_produs=v_id;
        DBMS_OUTPUT.put_line('Numarul de produse comandate pentru ' || v_nume || ' a fost actualizat la 6.');
    ELSE
```

```
DBMS_OUTPUT.put_line('Numarul de produse comandate pentru ' || v_nume || ' este mai mare de  
5.');
```

```
END IF;
```

```
CLOSE c;
```

```
EXCEPTION
```

```
WHEN produs_inexistent THEN
```

```
DBMS_OUTPUT.put_line('Nu exista id-ul  
introdus.');
```

```
END;
```

```
/
```

→id existent:

```
IF cNOTFOUND THEN
    RAISE produs_inexistent;
END IF;
IF v_numar<5 THEN
    UPDATE inventar
    SET numar_produse=6
    WHERE id_produs=v_id;
    DBMS_OUTPUT.put_line('Numarul de produse comandate pentru ' || v_nume || ' este mai mare de 5.');
ELSE
    DBMS_OUTPUT.put_line('Numarul de produse comandate pentru ' || v_nume || ' este mai mic de 5.');
END IF;
CLOSE c;
EXCEPTION
    WHEN produs_inexistent THEN
        DBMS_OUTPUT.put_line('Nu exista id-ul introdus.');
END;
```

PL/SQL procedure successfully completed.

→id nonexistent:

```
IF cNOTFOUND THEN
    RAISE produs_inexistent;
END IF;
IF v_numar<5 THEN
    UPDATE inventar
    SET numar_produse=6
    WHERE id_produs=v_id;
    DBMS_OUTPUT.put_line('Numarul de produse comandate pentru ' || v_nume || ' este mai mare de 5.');
ELSE
    DBMS_OUTPUT.put_line('Numarul de produse comandate pentru ' || v_nume || ' este mai mic de 5.');
END IF;
CLOSE c;
EXCEPTION
    WHEN produs_inexistent THEN
        DBMS_OUTPUT.put_line('Nu exista id-ul introdus.');
END;
```

PL/SQL procedure successfully completed.

- 
- \* Creează un bloc PL/SQL care să primească de la utilizator un oraș și să afișeze toate comenzi livrate către adresele din acel oraș. În cazul în care orașul introdus nu există în baza de date, afișează un mesaj corespunzător.

```
SET SERVEROUTPUT ON
ACCEPT oras PROMPT 'Introduceți orașul: '
DECLARE
    v_oras adrese.oras%TYPE := UPPER('&oras');
    oras_negasit EXCEPTION;
    CURSOR c IS
        SELECT adresa, id_comanda
        FROM adrese JOIN comenzi_restaurant USING (id_adresa)
        WHERE UPPER(oras) = v_oras;
    v c%ROWTYPE;
BEGIN
    OPEN c;
    FETCH c INTO v;
    IF c%NOTFOUND THEN
        CLOSE c;
        RAISE oras_negasit;
    END IF;
    LOOP
        DBMS_OUTPUT.PUT_LINE('Comanda cu id-ul ' || v.id_comanda || ' este livrata la adresa ' || v.adresa || '.');
        FETCH c INTO v;
        EXIT WHEN c%NOTFOUND;
    END LOOP;
    CLOSE c;
EXCEPTION
    WHEN oras_negasit THEN
```

```
DBMS_OUTPUT.PUT_LINE('Orasul introdus nu exista in baza de date.');
```

END;

/

→ oraș existent:

```
Worksheets | Query Builder
-----|-----
  FETCH c INTO v;
  IF c%NOTFOUND THEN
    CLOSE c;
    RAISE oras_negasit;
  END IF;
  LOOP
    DBMS_OUTPUT.PUT_LINE('Comanda cu id-ul ' || v.id_comanda || ' este
    FETCH c INTO v;
    EXIT WHEN c%NOTFOUND;
  END LOOP;
  CLOSE c;
EXCEPTION
  WHEN oras_negasit THEN
    DBMS_OUTPUT.PUT_LINE('Orasul introdus nu exista in baza de date.');
END;
/
-----|-----
Script Output | Task completed in 3.447 seconds
-----|-----
  END LOOP;
  CLOSE c;
EXCEPTION
  WHEN oras_negasit THEN
    DBMS_OUTPUT.PUT_LINE('Orasul introdus nu exista in baza de date.');
END;
Comanda cu id-ul C1001 este livrata la adresa Strada Victoriei, Nr. 1.
Comanda cu id-ul C1002 este livrata la adresa Bulevardul Unirii, Nr. 20.
Comanda cu id-ul C1008 este livrata la adresa Strada Independentei, Nr. 5.
Comanda cu id-ul C1010 este livrata la adresa Strada Stefan cel Mare, Nr. 30.
Comanda cu id-ul C1014 este livrata la adresa Bulevardul Timisara, Nr. 17.
```

→ oraș inexistent:

```
Worksheets | Query Builder
-----|-----
  FETCH c INTO v;
  IF c%NOTFOUND THEN
    CLOSE c;
    RAISE oras_negasit;
  END IF;
  LOOP
    DBMS_OUTPUT.PUT_LINE('Comanda cu id-ul ' || v.id_comanda || ' este
    FETCH c INTO v;
    EXIT WHEN c%NOTFOUND;
  END LOOP;
  CLOSE c;
EXCEPTION
  WHEN oras_negasit THEN
    DBMS_OUTPUT.PUT_LINE('Orasul introdus nu exista in baza de date.');
END;
/
-----|-----
Script Output | Task completed in 3.631 seconds
-----|-----
  LOOP
    DBMS_OUTPUT.PUT_LINE('Comanda cu id-ul ' || v.id_comanda || ' este livrat
    FETCH c INTO v;
    EXIT WHEN c%NOTFOUND;
  END LOOP;
  CLOSE c;
EXCEPTION
  WHEN oras_negasit THEN
    DBMS_OUTPUT.PUT_LINE('Orasul introdus nu exista in baza de date.');
END;
Orasul introdus nu exista in baza de date.
```

- 
- \* Scrie un bloc PL/SQL care să permită utilizatorului să introducă id-ul unui angajat. Blocul trebuie să afișeze numele angajatului și numărul total de schimburi efectuate de angajatul respectiv, în cazul în care angajatul există în baza de date. Dacă id-ul angajatului introdus de utilizator nu există, blocul trebuie să afișeze un mesaj corespunzător.

```
SET SERVEROUTPUT ON
ACCEPT id PROMPT 'Introduceți id-ul angajatului: '
DECLARE
    v_id rute.id_personal%TYPE:='&id';
    CURSOR c IS SELECT nume_personal, prenume_personal, COUNT(id_schimb) total_schimburi
        FROM personal JOIN rute USING (id_personal)
        WHERE id_personal = UPPER(v_id)
        GROUP BY nume_personal, prenume_personal;
    v c%ROWTYPE;
    angajat_inexistent EXCEPTION;
BEGIN
    OPEN c;
    FETCH c INTO v;
    IF c%NOTFOUND THEN
        CLOSE c;
        RAISE angajat_inexistent;
    END IF;
    DBMS_OUTPUT.PUT_LINE('Nume personal: ' || v.nume_personal || ' ' || v.prenume_personal);
    DBMS_OUTPUT.PUT_LINE('Numărul total de schimburi: ' || v.total_schimburi);
    CLOSE c;
EXCEPTION
    WHEN angajat_inexistent THEN
        DBMS_OUTPUT.PUT_LINE('Nu există date pentru angajatul specificat.');
    WHEN OTHERS THEN
```

```
DBMS_OUTPUT.PUT_LINE('A aparut o eroare.' || SQLERRM);
```

```
END;
```

```
/
```

→angajat existent:

The screenshot shows the Oracle SQL Developer interface with a worksheet tab open. The code is as follows:

```
OPEN c;
  FETCH c INTO v;
  IF c%NOTFOUND THEN
    CLOSE c;
    RAISE angajat_inexistent;
  END IF;
  DBMS_OUTPUT.PUT_LINE('Nume personal: ' || v.nume_personal || ' ');
  DBMS_OUTPUT.PUT_LINE('Numărul total de schimburi: ' || v.total_schimbur);
  CLOSE c;
EXCEPTION
  WHEN angajat_inexistent THEN
    DBMS_OUTPUT.PUT_LINE('Nu există date pentru angajatul specificat.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('A aparut o eroare.' || SQLERRM);
END;
```

The script output shows the results of the query:

```
DBMS_OUTPUT.PUT_LINE('Nume personal: ' || v.nume_personal || ' ');
DBMS_OUTPUT.PUT_LINE('Numărul total de schimburi: ' || v.total_schimbur);
CLOSE c;
EXCEPTION
  WHEN angajat_inexistent THEN
    DBMS_OUTPUT.PUT_LINE('Nu există date pentru angajatul specificat.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('A aparut o eroare.' || SQLERRM);
END;
Nume personal: Stancu Cristian
Numărul total de schimburi: 7
```

→angajat nonexistent:

The screenshot shows the Oracle SQL Developer interface with a worksheet tab open. The code is identical to the previous one but includes an additional END IF statement at the bottom.

```
OPEN c;
  FETCH c INTO v;
  IF c%NOTFOUND THEN
    CLOSE c;
    RAISE angajat_inexistent;
  END IF;
  DBMS_OUTPUT.PUT_LINE('Nume personal: ' || v.nume_personal || ' ');
  DBMS_OUTPUT.PUT_LINE('Numărul total de schimburi: ' || v.total_schimbur);
  CLOSE c;
EXCEPTION
  WHEN angajat_inexistent THEN
    DBMS_OUTPUT.PUT_LINE('Nu există date pentru angajatul specificat.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('A aparut o eroare.' || SQLERRM);
END;
```

The script output shows the results of the query:

```
END IF;
DBMS_OUTPUT.PUT_LINE('Nume personal: ' || v.nume_personal || ' ');
DBMS_OUTPUT.PUT_LINE('Numărul total de schimburi: ' || v.total_schimbur);
CLOSE c;
EXCEPTION
  WHEN angajat_inexistent THEN
    DBMS_OUTPUT.PUT_LINE('Nu există date pentru angajatul specificat.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('A aparut o eroare.' || SQLERRM);
END;
Nu există date pentru angajatul specificat.
```

- 
- \* Scrie un bloc PL/SQL care să afișeze prenumele unui client pe baza unui id introdus de utilizator. Verifică dacă prenumele clientului asociat id-ului introdus are o lungime mai mare de 5 caractere. Dacă lungimea prenumelui este mai mare de 5 caractere, afișează id-ul clientului și prenumele său, iar dacă nu, afișează un mesaj corespunzător.

```
SET SERVEROUTPUT ON
ACCEPT id PROMPT 'Introduceți id-ul clientului: '
DECLARE
    v_id clienti_restaurant.id_client%TYPE:=&id;
    v_prenume clienti_restaurant.prenume_client%TYPE;
BEGIN
    FOR client IN (SELECT prenume_client FROM clienti_restaurant WHERE id_client=v_id)
    LOOP
        v_prenume:=client.prenume_client;
        IF LENGTH(v_prenume)>5 THEN
            DBMS_OUTPUT.put_line('Clientul cu id-ul ' || v_id || ' are prenumele ' || v_prenume || '.');
        ELSE
            RAISE NO_DATA_FOUND;
        END IF;
    END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.put_line('Cliențul cu id-ul ' || v_id || ' nu are prenumele cu o lungime mai mare de 5 caractere.');
    END;
/

```

→ am introdus id-ul unui client care are lungimea prenumelui mai mare de 5 caractere:

```

Welcome Page Andra CLIENTI_RESTAURANT
Worksheet Query Builder
v_id clienti_restaurant.id_client%TYPE:=&id;
v_prenume clienti_restaurant.prenume_client%TYPE;
BEGIN
FOR client IN (SELECT prenume_client FROM clienti_restaurant WHERE
v_prenume:=client.prenume_client;
IF LENGTH(v_prenume)>5 THEN
DBMS_OUTPUT.put_line('Clientul cu id-ul ' || v_id || ' are prenumele ');
ELSE
RAISE NO_DATA_FOUND;
END IF;
END LOOP;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.put_line('Clinetul cu id-ul ' || v_id || ' nu are prenumele ');
END;

```

Script Output X | Task completed in 1.57 seconds

```

IF LENGTH(v_prenume)>5 THEN
DBMS_OUTPUT.put_line('Clientul cu id-ul ' || v_id || ' are prenumele ');
ELSE
RAISE NO_DATA_FOUND;
END IF;
END LOOP;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.put_line('Clinetul cu id-ul ' || v_id || ' nu are prenumele ');
END;
Clientul cu id-ul 8 are prenumele Cristian.

```

→ am introdus id-ul unui client care are lungimea prenumelui mai mică de 5 caractere:

```

Welcome Page Andra CLIENTI_RESTAURANT
Worksheet Query Builder
v_id clienti_restaurant.id_client%TYPE:=&id;
v_prenume clienti_restaurant.prenume_client%TYPE;
BEGIN
FOR client IN (SELECT prenume_client FROM clienti_restaurant WHERE
v_prenume:=client.prenume_client;
IF LENGTH(v_prenume)>5 THEN
DBMS_OUTPUT.put_line('Clientul cu id-ul ' || v_id || ' are prenumele ');
ELSE
RAISE NO_DATA_FOUND;
END IF;
END LOOP;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.put_line('Clinetul cu id-ul ' || v_id || ' nu are prenumele ');
END;

```

Script Output X | Task completed in 1.575 seconds

```

IF LENGTH(v_prenume)>5 THEN
DBMS_OUTPUT.put_line('Clientul cu id-ul ' || v_id || ' are prenumele ');
ELSE
RAISE NO_DATA_FOUND;
END IF;
END LOOP;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.put_line('Clinetul cu id-ul ' || v_id || ' nu are prenumele ');
END;
Clinetul cu id-ul 3 nu are prenumele cu o lungime mai mare de 5 caractere.

```

---

# PROCEDURI, FUNCȚII ȘI PACHETE

- \* Creează o procedură PL/SQL care să afișeze numărul total de rute pentru fiecare angajat într-un anumit interval de timp.

```
SET SERVEROUTPUT ON
```

```
CREATE OR REPLACE PROCEDURE numara_rute_pe_angajat(data_inceput IN DATE, data_sfarsit  
IN DATE)
```

```
IS
```

```
    total_rute NUMBER:=0;
```

```
    CURSOR c IS
```

```
        SELECT id_personal, COUNT(*) AS total_rute  
        FROM rute  
        WHERE data BETWEEN data_inceput AND data_sfarsit  
        GROUP BY id_personal;
```

```
    v_id_personal rute.id_personal%TYPE;
```

```
    total_rute_personal NUMBER;
```

```
    ex EXCEPTION;
```

```
BEGIN
```

```
    OPEN c;
```

```
    LOOP
```

```
        FETCH c INTO v_id_personal, total_rute_personal;
```

```
        EXIT WHEN c%NOTFOUND;
```

```
        DBMS_OUTPUT.PUT_LINE('Angajatul ' || v_id_personal || ' a avut ' || total_rute_personal || '  
        rute.');
```

```
        total_rute:=total_rute+total_rute_personal;
```

END LOOP;  
CLOSE c;

```
DBMS_OUTPUT.PUT_LINE('Numarul total de rute pentru intervalul ' || data_inceput || ' - ' || data_sfarsit || ' este: ' || total_rute);
```

IF total\_rute=0 THEN

RAISE ex;

END IF;

## EXCEPTION

## WHEN ex THEN

```
DBMS_OUTPUT.PUT_LINE('Nu s-au gasit rute pentru intervalul dat.');
```

END;

1

→ apelare:

```
EXECUTE numara_rute_pe_angajat(TO_DATE('01-12-2023', 'DD-MM-YYYY'), TO_DATE('15-12-2023', 'DD-MM-YYYY'));
```

1

- 
- \* Creează o procedură PL/SQL care să permită actualizarea cantității unui anumit ingredient în rețeta în care este utilizat.

```
CREATE OR REPLACE PROCEDURE actualizeaza_cantitate_ingredient_reteta(p_id_ingredient
IN VARCHAR2, p_noua_cantitate IN NUMBER)
```

```
IS
```

```
CURSOR c IS
```

```
    SELECT r.*, i.nume_ingredient  
    FROM retete r  
    JOIN ingrediente i ON r.id_ingredient=i.id_ingredient  
    WHERE r.id_ingredient=p_id_ingredient;
```

```
ingredient_negasit EXCEPTION;
```

```
v c%ROWTYPE;
```

```
BEGIN
```

```
OPEN c;
```

```
FETCH c INTO v;
```

```
IF c%NOTFOUND THEN
```

```
    CLOSE c;
```

```
    RAISE ingredient_negasit;
```

```
END IF;
```

```
CLOSE c;
```

```
UPDATE retete
```

```
SET cantitate_ingredient=p_noua_cantitate  
WHERE id_ingredient=p_id_ingredient;
```

```
DBMS_OUTPUT.PUT_LINE('Cantitatea ingredientului ' || v.nume_ingredient || ' a fost  
actualizata la ' || p_noua_cantitate || '' || v.unitate_de_masura || ' pentru reteta preparatului ' ||  
v.id_reteta || '.');
```

```
EXCEPTION
```

```

WHEN ingredient_negasit THEN
    DBMS_OUTPUT.PUT_LINE('Ingredientul specificat nu a fost gasit in nicio reteta.');
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('A intervenit o eroare in procesul de actualizare a cantitatii
ingredientului in reteta.');
END;
/
→apelare:
EXECUTE actualizeaza_cantitate_ingredient_reteta('I64', 3);
/

```

The screenshot shows the Oracle SQL Developer interface. The top menu bar includes 'Welcome Page', 'Andra', and various icons. Below the menu is a toolbar with icons for running scripts, saving, and zooming.

The main area has two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab contains the following PL/SQL code:

```

WORKSHEET - Query Builder
-----+
| FETCH c INTO v;
| IF c%NOTFOUND THEN
|   CLOSE c;
|   RAISE ingredient_negasit;
| END IF;
| CLOSE c;
|
| UPDATE retete
| SET cantitate_ingredient=p_noua_cantitate
| WHERE id_ingredient=p_id_ingredient;
|
| DBMS_OUTPUT.PUT_LINE('Cantitatea ingredientului ' || v.numar_ingredient || ' a fost actualizata la ' ||
|                      p_noua_cantitate);
|
| EXCEPTION
|   WHEN ingredient_negasit THEN
|     DBMS_OUTPUT.PUT_LINE('Ingredientul specificat nu a fost gasit in nicio reteta.');
|   WHEN OTHERS THEN
|     DBMS_OUTPUT.PUT_LINE('A intervenit o eroare in procesul de actualizare a cantitatii ingredientului
|                         in reteta.');
| END;
|
| EXECUTE actualizeaza_cantitate_ingredient_reteta('I64', 3);
|
| /
```

The 'Script Output' tab at the bottom shows the result of the execution:

```

Script Output x
Task completed in 0.148 seconds
Cantitatea ingredientului Rosie a fost actualizata la 3 buc pentru retata preparatului Sandwich Veggie.
```

- \* Creează o procedură PL/SQL pentru a calcula cantitatea totală comandată de produse.

```
CREATE OR REPLACE PROCEDURE cantitate_totala_comandata IS
```

```
    TYPE rec_produs IS RECORD (
        denumire_produs produse_restaurant.nume_produs%TYPE,
        cantitate_totala NUMBER
    );

```

```
    TYPE tab_produse IS TABLE OF rec_produs INDEX BY PLS_INTEGER;
```

```
    v_tab_produse tab_produse;
```

```
BEGIN
```

```
    SELECT nume_produs, NVL(SUM(numar_produse), 0)
    BULK COLLECT INTO v_tab_produse
    FROM produse_restaurant JOIN inventar USING (id_produs)
    JOIN comenzi_restaurant USING (id_comanda)
    GROUP BY nume_produs;
```

```
FOR i IN 1..v_tab_produse.COUNT LOOP
```

```
    DBMS_OUTPUT.PUT_LINE('Produsul ' ||
    v_tab_produse(i).denumire_produs || ' a fost
    comandat în cantitatea totală de ' ||
    v_tab_produse(i).cantitate_totala || ' buc.');
```

```
END LOOP;
```

```
END;
```

```
/
```

→ apelare:

```
EXECUTE cantitate_totala_comandata;
```

```
/
```

```
CREATE OR REPLACE PROCEDURE cantitate_totala_comandata IS
    TYPE rec_produs IS RECORD (
        denumire_produs produse_restaurant.nume_produs%TYPE,
        cantitate_totala NUMBER
    );
    TYPE tab_produse IS TABLE OF rec_produs INDEX BY PLS_INTEGER;
    v_tab_produse tab_produse;
BEGIN
    SELECT nume_produs, NVL(SUM(numar_produse), 0)
    BULK COLLECT INTO v_tab_produse
    FROM produse_restaurant JOIN inventar USING (id_produs)
    JOIN comenzi_restaurant USING (id_comanda)
    GROUP BY nume_produs;

    FOR i IN 1..v_tab_produse.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE('Produsul ' || v_tab_produse(i).denumire_produs || ' a fost
        comandat în cantitatea totală de ' ||
        v_tab_produse(i).cantitate_totala || ' buc.');
    END LOOP;
    EXECUTE cantitate_totala_comandata;
END;
```

Script Output X | Task completed in 0.154 seconds

Produsul Pasta Carbonara a fost comandat în cantitatea totală de 2 buc.  
 Produsul Paste Bolognese a fost comandat în cantitatea totală de 3 buc.  
 Produsul Pizza Margherita a fost comandat în cantitatea totală de 6 buc.  
 Produsul Pui la Gratar a fost comandat în cantitatea totală de 2 buc.  
 Produsul Supă de Legume a fost comandat în cantitatea totală de 3 buc.  
 Produsul Ghimbir Combo a fost comandat în cantitatea totală de 1 buc.

- 
- \* Creează o funcție PL/SQL de verificare a prețului produselor din restaurant. Utilizatorul va introduce id-ul unui produs, iar aceasta va afișa dacă prețul produsului este mai mare sau mai mic decât prețul mediu al tuturor produselor disponibile în restaurant. De asemenea, va afișa informații despre produsul respectiv, inclusiv numele, prețul și moneda.

```
CREATE OR REPLACE FUNCTION verifica_pretul(p_id_produs IN  
produse_restaurant.id_produs%TYPE, p_pret_mediul NUMBER)
```

```
RETURN BOOLEAN
```

```
IS
```

```
    v_pret produse_restaurant.pret_produs%TYPE;
```

```
BEGIN
```

```
    SELECT pret_produs INTO v_pret  
    FROM produse_restaurant  
    WHERE id_produs=p_id_produs;
```

```
    IF v_pret>p_pret_mediul THEN
```

```
        RETURN TRUE;
```

```
    ELSE
```

```
        RETURN FALSE;
```

```
    END IF;
```

```
EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
```

```
        RETURN NULL;
```

```
END;
```

```
/
```

```
CREATE OR REPLACE PROCEDURE calcul_pret_mediul(pret_mediul OUT NUMBER)
```

```
IS
```

```
BEGIN
```

```
SELECT AVG(pret_produs) INTO pret_mediul  
FROM produse_restaurant;
```

EXCEPTION

```
WHEN NO_DATA_FOUND THEN  
    pret_mediul:=NULL;
```

END;

/

→ apelare

```
ACCEPT id PROMPT 'Introduceti id-ul produsului: '
```

DECLARE

```
v_id produse_restaurant.id_produs%TYPE:=UPPER('&id');  
v_pret_mediul NUMBER;  
CURSOR c IS SELECT nume_produs, pret_produs, moneda  
    FROM produse_restaurant  
    WHERE id_produs=v_id;
```

```
v c%ROWTYPE;
```

BEGIN

```
calcul_pret_mediul(v_pret_mediul);  
DBMS_OUTPUT.put_line('Pretul mediul al produselor: ' || v_pret_mediul);
```

OPEN c;

LOOP

```
FETCH c INTO v;
```

```
EXIT WHEN c%NOTFOUND;
```

```
IF (verifica_pretul(v_id, v_pret_mediul) IS NULL) THEN  
    DBMS_OUTPUT.PUT_LINE('Produs cu ID invalid!');
```

```

ELSIF (verifica_pretul(v_id, v_pret_mediul)) THEN
    DBMS_OUTPUT.PUT_LINE('Produsul ' || v.nume_produs || ' cu id-ul ' || v_id || ' are pretul mai
mare decat pretul mediu, mai exact ' || v.pret_produs || '' || v.moneda || '.');
ELSE
    DBMS_OUTPUT.PUT_LINE('Produsul ' || v.nume_produs || ' cu id-ul ' || v_id || ' are pretul mai
mic decat pretul mediu, mai exact ' || v.pret_produs || '' || v.moneda || '.');
END IF;
END LOOP;
CLOSE c;
END;
/

```

The screenshot shows the Oracle SQL Developer interface. In the top window (Worksheet), a PL/SQL block is displayed. The code includes a cursor declaration, a BEGIN block with a LOOP, and various DBMS\_OUTPUT statements. The LOOP section is highlighted with a yellow background. In the bottom window (Script Output), the results of the execution are shown, including a message about a product ID being invalid and two lines of output regarding the price comparison for different products.

```

WHERE id_produs=v_id;
v c%ROWTYPE;
BEGIN
    calcul_pret_mediul(v_pret_mediul);
    DBMS_OUTPUT.put_line('Pretul mediu al produselor: ' || v_pret_mediul);

    OPEN c;
    LOOP
        FETCH c INTO v;
        EXIT WHEN c%NOTFOUND;
        IF (verifica_pretul(v_id, v_pret_mediul) IS NULL) THEN
            DBMS_OUTPUT.PUT_LINE('Produs cu ID invalid!');
        ELSIF (verifica_pretul(v_id, v_pret_mediul)) THEN
            DBMS_OUTPUT.PUT_LINE('Produsul ' || v.nume_produs || ' cu id-ul ' || v_id || ' are pretul mai mare decat pretul mediu, mai exact ' || v.pret_produs || '' || v.moneda || '.');
        ELSE
            DBMS_OUTPUT.PUT_LINE('Produsul ' || v.nume_produs || ' cu id-ul ' || v_id || ' are pretul mai mic decat pretul mediu, mai exact ' || v.pret_produs || '' || v.moneda || '.');
        END IF;
    END LOOP;
    CLOSE c;
END;
/

```

```

Script Output x
Task completed in 2.162 seconds
END LOOP;
CLOSE c;
END;
Pretul mediu al produselor: 14.417
Produsul Pizza Margherita cu id-ul P1 are pretul mai mare decat pretul mediu, mai exact 15.9 RON.

```

- 
- \* Creează o funcție PL/SQL care să actualizeze adresa din tabela adrese.  
Apelează această funcție într-un bloc PL/SQL care solicită utilizatorului să introducă id-ul adresei și noua adresă.

```
CREATE OR REPLACE FUNCTION modifica_adresa(p_id_adresa IN adrese.id_adresa%TYPE,  
p_noua_adresa IN adrese.adresa%TYPE)
```

```
RETURN BOOLEAN
```

```
IS
```

```
    v_count NUMBER;
```

```
BEGIN
```

```
    SELECT COUNT(*)
```

```
    INTO v_count
```

```
    FROM adrese
```

```
    WHERE id_adresa=p_id_adresa;
```

```
IF v_count>0 THEN
```

```
    UPDATE adrese
```

```
    SET adresa=p_noua_adresa
```

```
    WHERE id_adresa=p_id_adresa;
```

```
    RETURN TRUE;
```

```
ELSE
```

```
    RETURN FALSE;
```

```
END IF;
```

```
EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
```

```
        RETURN FALSE;
```

```
END;
```

```
/
```

```
→apelare
```

```
SET SERVEROUTPUT ON
```

ACCEPT id PROMPT 'Introduceți id-ul adresei (10-24):'

ACCEPT adresa PROMPT 'Introduceți noua adresa:'

DECLARE

```
v_id_adresa adrese.id_adresa%TYPE:=&id;
```

```
v_noua_adresa adrese.adresa%TYPE:='&adresa';
```

```
v_modificat BOOLEAN;
```

BEGIN

```
v_modificat:=modifica_adresa(v_id_adresa, v_noua_adresa);
```

IF v\_modificat THEN

```
    DBMS_OUTPUT.PUT_LINE('Adresa a fost modificata cu succes!');
```

ELSE

```
    DBMS_OUTPUT.PUT_LINE('Nu s-a putut modifica adresa sau id-ul adresei specificat nu  
exista!');
```

END IF;

END;

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, there are tabs for 'Welcome Page', 'Andra', and 'ADRESE'. Below the tabs, there are icons for file operations like New, Open, Save, and Print, along with search and help functions. The main area has two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab contains the PL/SQL code provided above. The 'Script Output' tab at the bottom shows the execution results:

```
Task completed in 42.602 seconds
DBMS_OUTPUT.PUT_LINE('Adresa a fost modificata cu succes!');
ELSE
    DBMS_OUTPUT.PUT_LINE('Nu s-a putut modifica adresa sau id-ul adresei specificat nu
exista!');
END IF;
END;
Adresa a fost modificata cu succes!
```

- 
- \* Creează o funcție PL/SQL care să calculeze media tarifelor pe oră pentru toți angajații dintr-o anumită poziție.

```
CREATE OR REPLACE FUNCTION calculeaza_medie_tarif(p_pozitie IN personal.pozitie%TYPE)
RETURN NUMBER
IS
    v_suma_tarife NUMBER:=0;
    v_numar_angajati NUMBER:=0;
    v_medie_tarif NUMBER;
    ex EXCEPTION;
BEGIN
    FOR rec IN (SELECT tarif_pe_ora FROM personal WHERE pozitie = p_pozitie) LOOP
        v_suma_tarife:=v_suma_tarife+rec.tarif_pe_ora;
        v_numar_angajati:=v_numar_angajati+1;
    END LOOP;

    IF v_numar_angajati=0 THEN
        RAISE ex;
    END IF;

    v_medie_tarif := v_suma_tarife / v_numar_angajati;
    RETURN v_medie_tarif;
EXCEPTION
    WHEN ex THEN
        DBMS_OUTPUT.put_line('Nu exista personal cu pozitia specificata!');
        RETURN NULL;
END;
/
```

→ apelare

DECLARE

v\_medie\_tarif NUMBER;

BEGIN

v\_medie\_tarif:=calculeaza\_medie\_tarif('Bucatar');

IF v\_medie\_tarif IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Media tarifelor pe ora pentru pozitia "Bucatar" este: ' ||  
v\_medie\_tarif);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Nu exista angajati cu pozitia "Bucatar".');

END IF;

END;

/

The screenshot shows the Oracle SQL Developer interface. The top menu bar includes 'Welcome Page', 'Andra', and 'ADRESE'. The main window has tabs for 'Worksheet' and 'Query Builder', with 'Worksheet' selected. The code area contains a PL/SQL block:

```
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.put_line('Nu exista personal cu pozitia specificata!');
    RETURN NULL;
  END;
/

DECLARE
  v_medie_tarif NUMBER;
BEGIN
  v_medie_tarif:=calculeaza_medie_tarif('Bucatar');

  IF v_medie_tarif IS NOT NULL THEN
    DBMS_OUTPUT.PUT_LINE('Media tarifelor pe ora pentru pozitia "Bucatar"
  ELSE
    DBMS_OUTPUT.PUT_LINE('Nu exista angajati cu pozitia "Bucatar".');
  END IF;
END;
/
```

The bottom part of the interface shows the 'Script Output' tab with the message: 'Task completed in 0.138 seconds'. It also displays the result of the function call: 'Function CALCULEAZA\_MEDIE\_TARIF compiled' and 'Media tarifelor pe ora pentru pozitia "Bucatar" este: 30'.

- \* Creează un pachet PL/SQL pentru gestionarea produselor într-un restaurant care să conțină:
  - procedura inserare\_produs trebuie să permită inserarea unui nou produs în tabelul produse\_restaurant;
  - funcția obtine\_pret\_produs trebuie să returneze prețul unui produs bazat pe id-ul produsului furnizat. Dacă id-ul produsului nu există în tabelul produse\_restaurant, funcția trebuie să returneze NULL;**Testează funcționalitatea pachetului.**

```
CREATE OR REPLACE PACKAGE pachet_produse_restaurant AS
```

```
  PROCEDURE inserare_produs(
    p_id_produs IN produse_restaurant.id_produs%TYPE,
    p_nume_produs IN produse_restaurant.nume_produs%TYPE,
    p_categorie_produs IN produse_restaurant.categorie_produs%TYPE,
    p_marime_produs IN produse_restaurant.marime_produs%TYPE,
    p_pret_produs IN produse_restaurant.pret_produs%TYPE
  );
```

```
  FUNCTION obtine_pret_produs(p_id_produs IN produse_restaurant.id_produs%TYPE) RETURN
NUMBER;
END;
/
```

```
CREATE OR REPLACE PACKAGE BODY pachet_produse_restaurant AS
```

```
  PROCEDURE inserare_produs(
    p_id_produs IN produse_restaurant.id_produs%TYPE,
    p_nume_produs IN produse_restaurant.nume_produs%TYPE,
    p_categorie_produs IN produse_restaurant.categorie_produs%TYPE,
    p_marime_produs IN produse_restaurant.marime_produs%TYPE,
    p_pret_produs IN produse_restaurant.pret_produs%TYPE
  ) IS
  BEGIN
    INSERT INTO produse_restaurant (id_produs, nume_produs, categorie_produs, marime_produs,
    pret_produs)
    VALUES (p_id_produs, p_nume_produs, p_categorie_produs, p_marime_produs,
    p_pret_produs);
    DBMS_OUTPUT.PUT_LINE('Produsul ' || p_nume_produs || ' a fost inserat cu succes.');
  END;
```

```

FUNCTION obtine_pret_produs(p_id_produs IN produse_restaurant.id_produs%TYPE) RETURN
NUMBER IS
    v_pret NUMBER;
BEGIN
    SELECT pret_produs INTO v_pret
    FROM produse_restaurant
    WHERE id_produs=p_id_produs;

    RETURN v_pret;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
END;
END;
/

```

```

BEGIN
    pachet_produse_restaurant.inserare_produs('P16', 'Ciorba Radauteana', 'Ciorba', 'portie mare', 15.5);
END;
/

```

```

DECLARE
    v_pret NUMBER;
BEGIN
    v_pret:=pachet_produse_restaurant.obtine_pret_produs('P16');
    IF v_pret IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('Preul produsului este: ' || v_pret);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Produsul nu a fost gasit.');
    END IF;
END;
/

```

The screenshot shows the Oracle SQL Developer interface with the following details:

- Worksheet Tab:** Contains the PL/SQL code for the package body.
- Script Output Tab:** Shows the results of the execution, including the successful compilation of the package and the insertion of a product record.

```

-- Worksheet Tab Content --
END;
/
BEGIN
    pachet_produse_restaurant.inserare_produs('P16', 'Ciorba Radauteana', 'Ciorba', 'portie mare', 15.5);
END;
/

```

```

-- Script Output Tab Content --
Package PACHET_PRODUSE_RESTAURANT compiled

Package Body PACHET_PRODUSE_RESTAURANT compiled
produsul Ciorba Radauteana a fost inserat cu succes.

PL/SQL procedure successfully completed.

retul produsului este: 15.5

PL/SQL procedure successfully completed.

```

- \* Creează un pachet PL/SQL care să se denumească pachet\_adrese care și să furnizeze funcționalități pentru actualizarea și obținerea detaliilor adresei. Acesta trebuie să conțină o procedură pentru actualizarea adresei și o funcție pentru obținerea detaliilor adresei pe baza id-ului adresei furnizat.

```

CREATE OR REPLACE PACKAGE pachet_adrese AS
  PROCEDURE actualizare_adresa(p_id_adresa IN adrese.id_adresa%TYPE, p_noua_adresa IN
adrese.adresa%TYPE);
  FUNCTION detalii_adresa(p_id_adresa IN adrese.id_adresa%TYPE) RETURN VARCHAR2;
END;
/

```

```

CREATE OR REPLACE PACKAGE BODY pachet_adrese AS
  PROCEDURE actualizare_adresa(p_id_adresa IN adrese.id_adresa%TYPE, p_noua_adresa IN
adrese.adresa%TYPE) IS
    BEGIN
      UPDATE adrese
      SET adresa=p_noua_adresa
      WHERE id_adresa=p_id_adresa;

      DBMS_OUTPUT.PUT_LINE('Adresa cu id-ul ' || p_id_adresa || ' a fost actualizata cu succes.');
    END;

  FUNCTION detalii_adresa(p_id_adresa IN adrese.id_adresa%TYPE) RETURN VARCHAR2 IS
    v_detalii VARCHAR2(200);
  BEGIN
    SELECT adresa || ',' || oras || ',' || cod_postal INTO v_detalii
    FROM adrese
    WHERE id_adresa=p_id_adresa;

    RETURN v_detalii;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      RETURN NULL;
  END;
END;
/
BEGIN
  pachet_adrese.actualizare_adresa('25', 'Strada Anton Pann, Nr.14');
END;
/

```

```

DECLARE
    v_detalii_adresa VARCHAR2(200);
BEGIN
    v_detalii_adresa := pachet_adrese.detalii_adresa('25');
    IF v_detalii_adresa IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('Detalii adresa: ' || v_detalii_adresa);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Nu s-au gasit detalii pentru adresa specificata.');
    END IF;
END;
/

```

```

Welcome Page Andra [ADRESE]
Worksheet Query Builder
/
BEGIN
    pachet_adrese.actualizare_adresa('25', 'Strada Anton Pann, Nr.14');
END;
/

DECLARE
    v_detalii_adresa VARCHAR2(200);
BEGIN
    v_detalii_adresa := pachet_adrese.detalii_adresa('25');
    IF v_detalii_adresa IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('Detalii adresa: ' || v_detalii_adresa);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Nu s-au gasit detalii pentru adresa specificata.');
    END IF;
END;
/
Script Output X
Task completed in 0.236 seconds
Package PACHET_ADRESE compiled

Package Body PACHET_ADRESE compiled

Adresa cu id-ul 25 a fost actualizata cu succes.

PL/SQL procedure successfully completed.

Detalii adresa: Strada Anton Pann, Nr.14, Bucuresti, 987634

PL/SQL procedure successfully completed.

```

---

# DECLANŞATORI

- \* Implementează un trigger care să fie declanșat înainte de inserarea sau actualizarea în tabela ingrediente. Trigger-ul ar trebui să valideze cantitatea introdusă sau actualizată pentru fiecare înregistrare nouă sau actualizată. Dacă cantitatea este negativă, trigger-ul ar trebui să genereze o excepție cu un mesaj corespunzător. Testează pentru a garanta că doar cantități valide sunt introduse sau actualizate în tabela ingrediente.

```
CREATE OR REPLACE TRIGGER trg_ingrediente_actiuni
BEFORE INSERT OR DELETE OR UPDATE ON ingrediente
FOR EACH ROW
DECLARE
    v_cantitate_disponibila NUMBER;
BEGIN
    IF INSERTING THEN
        IF :NEW.cantitate>=0 THEN
            DBMS_OUTPUT.put_line('Cantitatea introdusa este valida.');
        ELSE
            RAISE_APPLICATION_ERROR(-20001, 'Cantitatea pe care doresti sa o introduci trebuie sa fie pozitiva.');
        END IF;
    ELSIF UPDATING THEN
        IF :NEW.cantitate>=0 THEN
            DBMS_OUTPUT.put_line('Cantitatea actualizata este valida.');
        ELSE
            RAISE_APPLICATION_ERROR(-200002, 'Cantitatea pe care doresti sa o actualizezi trebuie sa fie pozitiva.');
        END IF;
```

```

END IF;

END;

/

```

INSERT INTO ingrediente VALUES('I74', 'Faina', 10, 'kg');

```

UPDATE ingrediente
SET cantitate=20
WHERE id_ingredient='I74';

```

The screenshot shows the Oracle SQL Developer interface. The top part displays a trigger creation script in the 'Query Builder' tab:

```

ELSIF UPDATING THEN
    IF :NEW.cantitate>=0 THEN
        DBMS_OUTPUT.put_line('Cantitatea actualizata este valida.');
    ELSE
        RAISE_APPLICATION_ERROR(-200002, 'Cantitatea pe care doresti sa o actualizezi trebuie sa fie pozitiva.');
    END IF;
END IF;
END;
/

INSERT INTO ingrediente VALUES('I74', 'Faina', 10, 'kg');

UPDATE ingrediente
SET cantitate=20
WHERE id_ingredient='I74';

```

The bottom part shows the 'Script Output' tab with the results of the execution:

```

Task completed in 0.135 seconds

Trigger TRG_INGREDIENTE_ACTIUNI compiled

Cantitatea introdusa este valida.

1 row inserted.

Cantitatea actualizata este valida.

1 row updated.

```

- \* Implementează un trigger pentru tabela produse\_restaurant care să detecteze modificările semnificative de preț ale produselor. Trigger-ul trebuie să afișeze un mesaj atunci când prețul unui produs este modificat cu mai mult de 10%. Asigură-te că trigger-ul funcționează corect și că mesajele sunt afișate corespunzător în cazul modificărilor de preț.

```
CREATE OR REPLACE TRIGGER trg_modificari_pret
```

```
FOR UPDATE OF pret_produs ON produse_restaurant
```

```
COMPOUND TRIGGER
```

```
v_pret_vechi NUMBER;
```

```
BEFORE EACH ROW IS
```

```
BEGIN
```

```
    v_pret_vechi:=:OLD.pret_produs;
```

```
END BEFORE EACH ROW;
```

```
AFTER EACH ROW IS
```

```
BEGIN
```

```
    IF ABS(:NEW.pret_produs-v_pret_vechi)/v_pret_vechi>0.1 THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Modificare de pret semnificativa pentru produsul ' ||  
:NEW.nume_produs || '.');
```

```
    END IF;
```

```
END AFTER EACH ROW;
```

```
END;
```

```
/
```

```
UPDATE produse_restaurant
```

```
SET pret_produs=pret_produs*1.15
```

```
WHERE id_produs='P2';
```

The screenshot shows the Oracle SQL Developer interface. The top part displays the SQL code for creating the trigger:

```
CREATE OR REPLACE TRIGGER trg_modificari_pret
FOR UPDATE OF pret_produs ON produse_restaurant
COMPOUND TRIGGER
    v_pret_vechi NUMBER;
    BEFORE EACH ROW IS
    BEGIN
        v_pret_vechi:=:OLD.pret_produs;
    END BEFORE EACH ROW;

    AFTER EACH ROW IS
    BEGIN
        IF ABS(:NEW.pret_produs-v_pret_vechi)/v_pret_vechi>0.1 THEN
            DBMS_OUTPUT.PUT_LINE('Modificare de pret semnificativa pentru produsul ' || :NEW.nume_produs || '.');
        END IF;
    END AFTER EACH ROW;
END;
/
UPDATE produse_restaurant
SET pret_produs=pret_produs*1.15
WHERE id_produs='P2';
```

The bottom part shows the 'Script Output' window with the results of the trigger execution:

```
Task completed in 0.223 seconds
Trigger TRG_MODIFICARI_PRET compiled
Modificare de pret semnificativa pentru produsul Paste Bolognese.

now updated.
```

- \* Creează un trigger care să se activeze înainte de actualizarea tarifului pe oră pentru angajații din tabela personal. Acesta trebuie să rețină poziția anterioară a angajatului înainte de actualizare și să verifice dacă tariful pe oră a fost modificat. În caz afirmativ, trigger-ul trebuie să afișeze un mesaj care să indice că tariful a fost actualizat pentru angajatul respectiv.

```
CREATE OR REPLACE TRIGGER trg_personal_update
```

```
BEFORE UPDATE OF tarif_pe_ora ON personal
```

```
FOR EACH ROW
```

```
DECLARE
```

```
    v_pozitie_anterioara VARCHAR2(30);
```

```
BEGIN
```

```
    v_pozitie_anterioara:=:OLD.pozitie;
```

```
    IF :NEW.tarif_pe_ora<>:OLD.tarif_pe_ora THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Tariful pe ora a fost actualizat pentru angajatul cu id-ul: ' ||  
        :NEW.id_personal || '.');
```

```
    END IF;
```

```
END;
```

```
/
```

```
UPDATE personal
```

```
SET tarif_pe_ora=tarif_pe_ora*1.1
```

```
WHERE id_personal='ANG1';
```

The screenshot shows the Oracle SQL Developer interface. In the top-left corner, there's a small icon of a person with a crown, which is the logo for the Personal Development section. The main area displays the following PL/SQL code:

```

SET pret_produs=pret_produs*1.15
WHERE id_produs='P2';
/

CREATE OR REPLACE TRIGGER trg_personal_update
BEFORE UPDATE OF tarif_pe_ora ON personal
FOR EACH ROW
DECLARE
    v_pozitie_anterioara VARCHAR2(30);
BEGIN
    v_pozitie_anterioara:=:OLD.pozitie;
    IF :NEW.tarif_pe_ora<>:OLD.tarif_pe_ora THEN
        DBMS_OUTPUT.PUT_LINE('Tariful pe ora a fost actualizat pentru angajatul cu id-ul: ' || :NEW.id_personal || '.');
    END IF;
END;
/

```

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

```

Trigger TRG_PERSONAL_UPDATE compiled

Tariful pe ora a fost actualizat pentru angajatul cu id-ul: ANG1.

row updated.

```

- 
- \* Implementează un trigger în tabela adrese care să prevină inserarea unei adrese noi în cazul în care există comenzi asociate acelei adrese. De asemenea, acesta ar trebui să ridice o excepție dacă se încearcă inserarea unei adrese fără specificarea unui ID.

```
CREATE OR REPLACE TRIGGER trg_adrese_insert
```

```
BEFORE INSERT ON adrese
```

```
FOR EACH ROW
```

```
DECLARE
```

```
    v_numar_comenzi NUMBER;
```

```
BEGIN
```

```
    SELECT COUNT(*)
```

```
    INTO v_numar_comenzi
```

```
    FROM comenzi_restaurant
```

```
    WHERE id_adresa=:OLD.id_adresa;
```

```
    IF v_numar_comenzi>0 THEN
```

```
        RAISE_APPLICATION_ERROR(-20001, 'Nu se poate adauga adresa. Există comenzi asociate.');
```

```
    END IF;
```

```
    IF :NEW.id_adresa IS NULL THEN
```

```
        RAISE_APPLICATION_ERROR(-20002, 'Nu se poate adauga adresa fără specificarea unui ID.');
```

```
    END IF;
```

```
END;
```

```
/          Trigger TRG_ADRESE_INSERT compiled
```

```
INSERT INTO adrese VALUES (25, 'Strada Avram Iancu, Nr. 13', 'Bucuresti', 987634);
```

```
1 row inserted.
```