



ACADEMIA DE STUDII ECONOMICE  
FACULTATEA DE CIBERNETICĂ, STATISTICĂ ȘI INFORMATICĂ ECONOMICĂ

# **Proiect baze de date**

## **Gestiunea unui lanț de cafenele**

**Profesor coordonator:**  
Andreea Vines

**Proiect realizat de studenta:**  
Mocanu Valentina-Adriana

An universitar 2022-2023

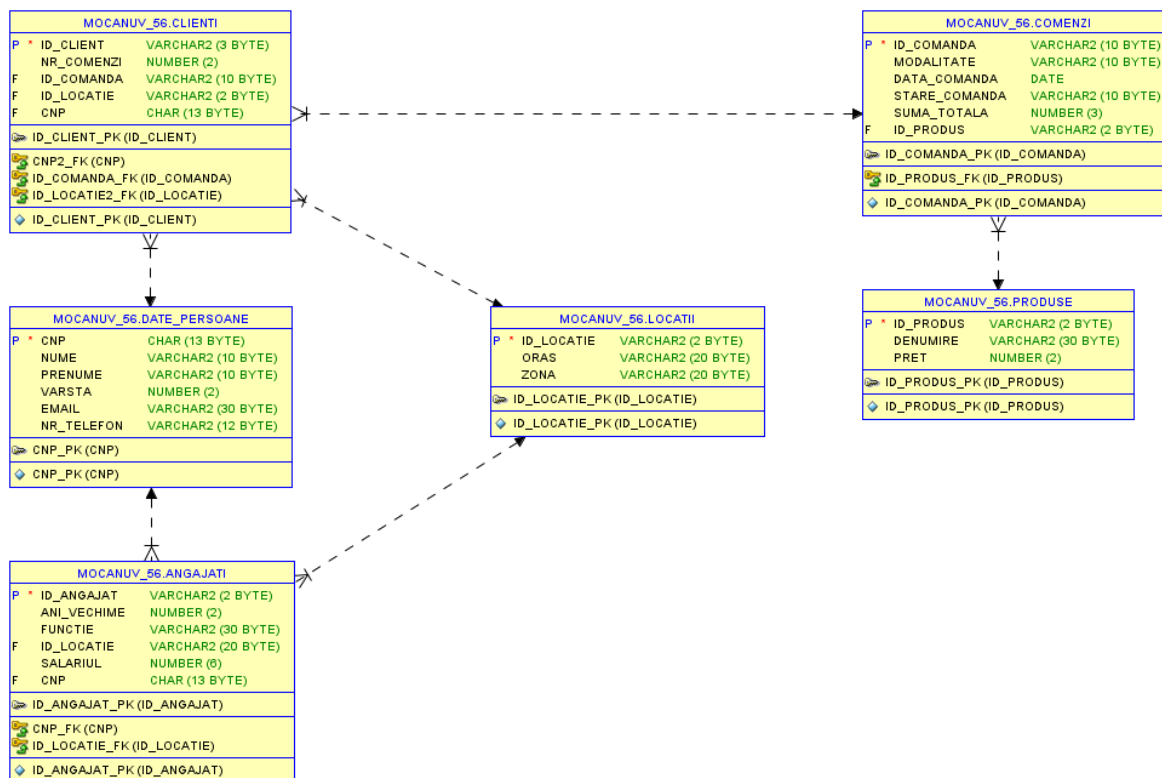
## A.Obiectivul proiectului

Această bază de date centralizează și supraveghează datele legate de administrarea unui lanț de cafenele. În cadrul acestui proiect se urmărește monitorizarea vânzărilor și a profitului în cadrul francizei. Prin intermediul bazei de date se vor stoca astfel următoarele informații:

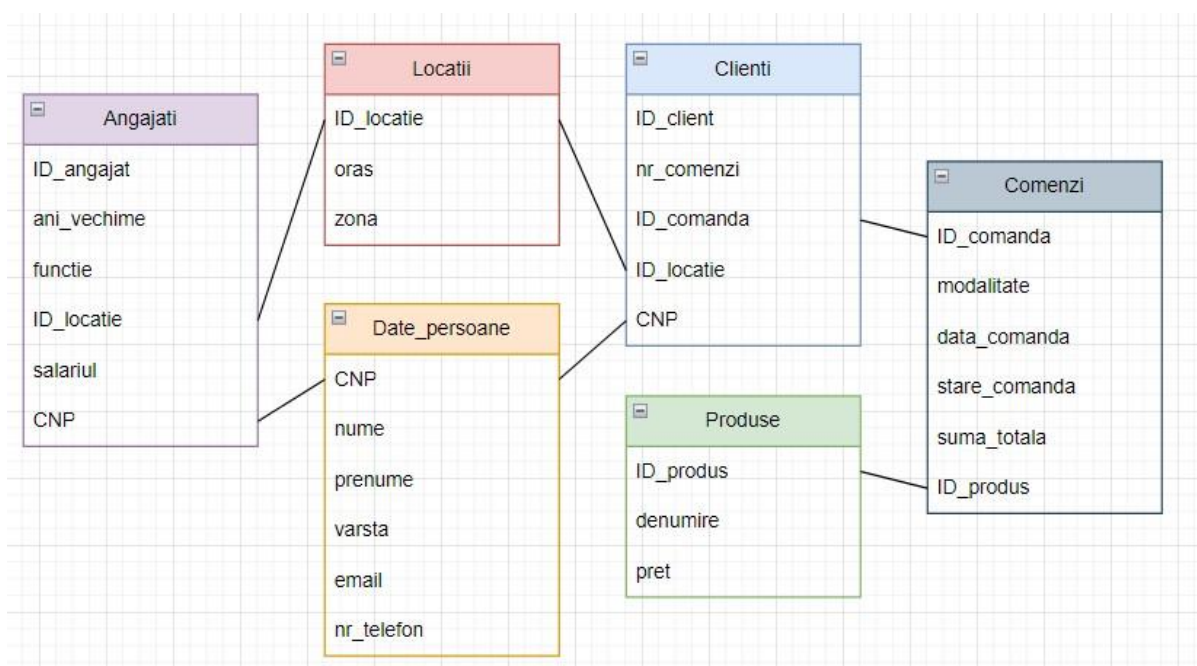
- datele atât personale, cât și de contact ale angajaților și ale clienților;
- date privind comenzile făcute de clienți pentru facilitarea livrării acestora;
- gestionarea produselor expuse și a prețurilor acestora.

Denumire Tabela	Atribute	Tip Data	Restrictii
Angajati	ID_angajat ani_vechime functie ID_locatie salariul CNP	VARCHAR2(2) NUMBER(2) VARCHAR2(30) VARCHAR2(20) NUMBER(6) CHAR (13)	PK   FK  FK
Locatii	ID_locatie oras zona	VARCHAR2(2) VARCHAR2(20) VARCHAR2(20)	PK
Cienti	ID_client nr_comenzi ID_comanda ID_locatie CNP	VARCHAR2(3) NUMBER(2) VARCHAR2(10) VARCHAR2(2) CHAR(13)	PK  FK FK FK
Date_persoane	CNP nume prenume varsta email nr_telefon	CHAR(13) VARCHAR2(10) VARCHAR2(10) NUMBER(2) VARCHAR2(30) VARCHAR2(12)	PK
Comenzi	ID_comanda modalitate data_comanda stare_comanda suma_totala ID_produs	VARCHAR2(10) VARCHAR2(15) DATE VARCHAR2(10) NUMBER(3) VARCHAR2(2)	PK    FK
Produse	ID_produs denumire pret	VARCHAR2(2) VARCHAR2(30) NUMBER(2)	PK

## Tipuri de legături



## Schema bazei de date



## Crearea tabelelor

```
CREATE TABLE Date_persoane
```

```
(  
    CNP CHAR(13),  
    nume VARCHAR2(10),  
    prenume VARCHAR2(10),  
    varsta NUMBER(2),  
    email VARCHAR2(30),  
    nr_telefon VARCHAR2(12)  
);
```

```
ALTER TABLE Date_persoane
```

```
ADD CONSTRAINT CNP_PK PRIMARY KEY (CNP);
```

```
CREATE TABLE Locatii
```

```
(  
    ID_locatie VARCHAR2(2),  
    oras VARCHAR2(20),  
    zona VARCHAR2(20)  
);
```

```
ALTER TABLE Locatii
```

```
ADD CONSTRAINT ID_LOCATIE_PK PRIMARY KEY (ID_locatie);
```

```
CREATE TABLE Prognose
```

```
(  
    ID_produs VARCHAR2(2),  
    denumire VARCHAR2(30),
```

```
pret NUMBER(2)
);
```

```
ALTER TABLE Produce
ADD CONSTRAINT ID_PRODUS_PK PRIMARY KEY (ID_produs);
```

```
CREATE TABLE Angajati
(
    ID_angajat VARCHAR2(2),
    ani_vechime NUMBER(2),
    functie VARCHAR2(30),
    ID_locatie VARCHAR2(20),
    salariul NUMBER(6),
    CNP CHAR(13)
);
```

```
ALTER TABLE Angajati
ADD CONSTRAINT ID_ANGAJAT_PK PRIMARY KEY (ID_angajat);
ALTER TABLE Angajati
ADD CONSTRAINT ID_LOCATIE_FK FOREIGN KEY (ID_locatie) REFERENCES Locatii
(ID_locatie);
ALTER TABLE Angajati
ADD CONSTRAINT CNP_FK FOREIGN KEY (CNP) REFERENCES Date_persoane (CNP);
```

```
CREATE TABLE Comenzi
(
    ID_comanda VARCHAR2(10),
    modalitate VARCHAR2(10),
    data_comanda DATE,
    stare_comanda VARCHAR2(10),
    suma_totala NUMBER(3),
```

```
ID_produș VARCHAR2(2)
);
```

```
ALTER TABLE Comenzi
ADD CONSTRAINT ID_COMANDA_PK PRIMARY KEY (ID_comanda);
ALTER TABLE Comenzi
ADD CONSTRAINT ID_PRODUS_FK FOREIGN KEY (ID_produș) REFERENCES Produce
(ID_produș);
```

```
CREATE TABLE Clienti
(
    ID_client VARCHAR2(3),
    nr_comenzi NUMBER(2),
    ID_comanda VARCHAR2(10),
    ID_locatie VARCHAR2(2),
    CNP CHAR(13)
);
```

```
ALTER TABLE Clienti
ADD CONSTRAINT ID_CLIENT_PK PRIMARY KEY (ID_client);
ALTER TABLE Clienti
ADD CONSTRAINT ID_COMANDA_FK FOREIGN KEY (ID_comanda) REFERENCES Comenzi
(ID_comanda);
ALTER TABLE Clienti
ADD CONSTRAINT ID_LOCATIE2_FK FOREIGN KEY (ID_locatie) REFERENCES Locatii
(ID_locatie);
ALTER TABLE Clienti
ADD CONSTRAINT CNP2_FK FOREIGN KEY (CNP) REFERENCES Date_persoane (CNP);
```

Columns   Data   Model   Constraints   Grants   Statistics   Triggers   Flashback   Dependencies   Details   Partitions   Indexes   SQL						
<div>    Actions... </div>						
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_ANGAJAT	VARCHAR2(2 BYTE)	No	(null)	1 (null)	
2	ANI_VECHIME	NUMBER(2,0)	Yes	(null)	2 (null)	
3	FUNCTIE	VARCHAR2(30 BYTE)	Yes	(null)	3 (null)	
4	ID_LOCATIE	VARCHAR2(20 BYTE)	Yes	(null)	4 (null)	
5	CNP	CHAR(13 BYTE)	Yes	(null)	5 (null)	

Columns   Data   Model   Constraints   Grants   Statistics   Triggers   Flashback   Dependencies   Details   Partitions   Indexes   SQL						
<div>    Actions... </div>						
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_CLIENT	VARCHAR2(3 BYTE)	No	(null)	1 (null)	
2	NR_COMENZI	NUMBER(2,0)	Yes	(null)	2 (null)	
3	ID_COMANDA	VARCHAR2(10 BYTE)	Yes	(null)	3 (null)	
4	ID_LOCATIE	VARCHAR2(2 BYTE)	Yes	(null)	4 (null)	
5	CNP	CHAR(13 BYTE)	Yes	(null)	5 (null)	

Columns   Data   Model   Constraints   Grants   Statistics   Triggers   Flashback   Dependencies   Details   Partitions   Indexes   SQL						
<div>    Actions... </div>						
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_COMANDA	VARCHAR2(10 BYTE)	No	(null)	1 (null)	
2	MODALITATE	VARCHAR2(10 BYTE)	Yes	(null)	2 (null)	
3	DATA_COMANDA	DATE	Yes	(null)	3 (null)	
4	STARE_COMANDA	VARCHAR2(10 BYTE)	Yes	(null)	4 (null)	
5	SUMA_TOTALA	NUMBER(3,0)	Yes	(null)	5 (null)	
6	ID_PRODUS	VARCHAR2(2 BYTE)	Yes	(null)	6 (null)	

Columns   Data   Model   Constraints   Grants   Statistics   Triggers   Flashback   Dependencies   Details   Partitions   Indexes   SQL						
<div>    Actions... </div>						
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	CNP	CHAR(13 BYTE)	No	(null)	1 (null)	
2	NUME	VARCHAR2(10 BYTE)	Yes	(null)	2 (null)	
3	PRENUME	VARCHAR2(10 BYTE)	Yes	(null)	3 (null)	
4	VARSTA	NUMBER(2,0)	Yes	(null)	4 (null)	
5	EMAIL	VARCHAR2(30 BYTE)	Yes	(null)	5 (null)	
6	NR_TELEFON	VARCHAR2(12 BYTE)	Yes	(null)	6 (null)	

Adriana x LOCATII x						
Columns   Data   Model   Constraints   Grants   Statistics   Triggers   Flashback   Dependencies   Details   Partitions   Indexes						
Actions...						
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_LOCATIE	VARCHAR2 (2 BYTE)	No	(null)	1 (null)	
2	ORAS	VARCHAR2 (20 BYTE)	Yes	(null)	2 (null)	
3	ZONA	VARCHAR2 (20 BYTE)	Yes	(null)	3 (null)	

Adriana x PRODUSE x						
Columns   Data   Model   Constraints   Grants   Statistics   Triggers   Flashback   Dependencies   Details   Partitions   Indexes						
Actions...						
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_PRODUS	VARCHAR2 (2 BYTE)	No	(null)	1 (null)	
2	DENUMIRE	VARCHAR2 (30 BYTE)	Yes	(null)	2 (null)	
3	PRET	NUMBER (2, 0)	Yes	(null)	3 (null)	

## Adăugarea de înregistrări în fiecare tabelă

### 1. Tabela Date\_persoane

DECLARE

TYPE cnp\_arr IS TABLE OF VARCHAR2(13);

TYPE nume\_arr IS TABLE OF VARCHAR2(50);

TYPE prenume\_arr IS TABLE OF VARCHAR2(50);

TYPE varsta\_arr IS TABLE OF NUMBER;

TYPE email\_arr IS TABLE OF VARCHAR2(100);

TYPE nr\_telefon\_arr IS TABLE OF VARCHAR2(15);

v\_cnp cnp\_arr := cnp\_arr(

'5230103407167', '6235253404967', '6232903474193', '5239371628367', '6468453132613',  
 '6237591238456', '5234891576236', '6280305727918', '5120307918573', '5187629438159',  
 '5260345182976', '6280418273619', '5482916732840', '6182937581916', '6618297329162',  
 '5190602182719', '6050203182947', '6281937482956', '5712973481952', '6290718396281',  
 '5290506285349'

);

v\_nume nume\_arr := nume\_arr(



```
'Popescu', 'Lazar', 'Anghel', 'Nicolae', 'Mares',  
'Mocanu', 'Mocanu', 'Marinescu', 'Tudor', 'Tudoroiu',  
'Mihai', 'Neamtu', 'Moise', 'Manu', 'Marin',  
'Militaru', 'Ionescu', 'Enache', 'Elcu', 'Dinca',  
'Neacsu'  
);
```

```
v_prenume prenume_arr := prenume_arr(  
  'Ion', 'Adelina', 'Georgiana', 'Mircea', 'Maria',  
  'Luna', 'Merlin', 'Ioana', 'Valentin', 'Eduard',  
  'Adrian', 'Cristina', 'Alexandra', 'Dan', 'Iustin',  
  'Mihaela', 'Pavel', 'Radu', 'Lorena', 'Alin',  
  'Andrei'  
);
```

```
v_varsta varsta_arr := varsta_arr(  
  15, 21, 18, 15, 20,  
  21, 22, 30, 35, 22,  
  46, 25, 34, 50, 14,  
  20, 23, 24, 17, 40,  
  18  
);
```

```
v_email email_arr := email_arr(  
  'pion15@yahoo.com', 'ladelina@gmail.com', 'ageorgiana@gmail.com', 'nmircea@yahoo.com',  
  'mmaria@gmail.com',  
  'luna@gmail.com', 'merlin@gmail.com', 'mioana@yahoo.com', 'tvalentin@gmail.com',  
  'teduard@gmail.com',  
  'madrian@gmail.com', 'ncristina@yahoo.com', 'malexandra@gmail.com', 'mdan@gmail.com',  
  'miustin@yahoo.com',  
  'mmihaela@gmail.com', 'ipavel@gmail.com', 'eradu@yahoo.com', 'elorena@gmail.com',  
  'dalin@gmail.com',  
  'nandrei@yahoo.com'  
);
```

```

v_nr_telefon nr_telefon_arr := nr_telefon_arr(

'+40742830285', '+40748371985', '+40742934713', '+40725893478', '+40758201479',

'+40743921678', '+40742539182', '+40752104983', '+40743859217', '+40751293657',

'+40758394712', '+40729837592', '+40756819374', '+40751203846', '+40738192057',

'+40750293741', '+40728461937', '+40736529401', '+40742916374', '+40718492653',

'+40739182746'

);

BEGIN

FOR i IN 1..v_cnp.COUNT LOOP

INSERT INTO Date_persoane (CNP, nume, prenume, varsta, email, nr_telefon)

VALUES (v_cnp(i), v_nume(i), v_prenume(i), v_varsta(i), v_email(i), v_nr_telefon(i));

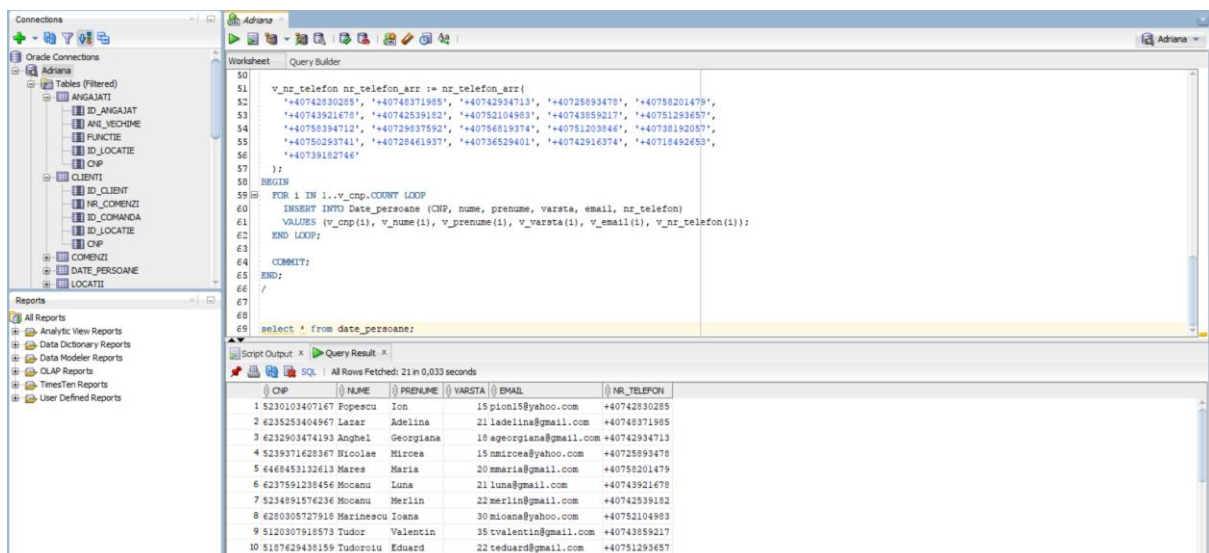
END LOOP;

COMMIT;

END;

/

```



The screenshot shows the Oracle SQL Developer interface. The 'Query Builder' window displays the following SQL script:

```

50 v_nr_telefon nr_telefon_arr := nr_telefon_arr(
51 '+40742830285', '+40748371985', '+40742934713', '+40725893478', '+40758201479',
52 '+40743921678', '+40742539182', '+40752104983', '+40743859217', '+40751293657',
53 '+40758394712', '+40729837592', '+40756819374', '+40751203846', '+40738192057',
54 '+40750293741', '+40728461937', '+40736529401', '+40742916374', '+40718492653',
55 '+40739182746'
56 );
57
58 BEGIN
59 FOR i IN 1..v_cnp.COUNT LOOP
60 INSERT INTO Date_persoane (CNP, nume, prenume, varsta, email, nr_telefon)
61 VALUES (v_cnp(i), v_nume(i), v_prenume(i), v_varsta(i), v_email(i), v_nr_telefon(i));
62 END LOOP;
63
64 COMMIT;
65 END;
66 /
67
68 select * from date_persoane;
69

```

The 'Query Result' window shows the following data:

CNP	NUME	PRENUME	VARSTA	EMAIL	NR_TELEFON
1 5230103407167	Popescu	Ion	15	ipion1@yahoo.com	+40742830285
2 4235283404967	Leasar	Adelina	21	ladelina@gmail.com	+40748371985
3 4232903474193	Angelhel	Georgiana	18	ageorgiana@gmail.com	+40742934713
4 5239371628367	Nicolae	Mirocea	15	mmirocea@yahoo.com	+40725893478
5 4468453132613	Marex	Maria	20	mmaria@gmail.com	+40758201479
6 4237591239456	Mocanu	Luna	21	luna@gmail.com	+40743921678
7 5234891574236	Mocanu	Merlin	22	merlin@gmail.com	+40742539182
8 4280305727916	Marinescu	Ioana	30	miocnea@yahoo.com	+40752104983
9 5120307918573	Tudor	Valentin	35	vvalentin@gmail.com	+40743859217
10 5187429438159	Tudoroiu	Eduard	22	teduard@gmail.com	+40751293657

## 2. Tabela Locatii

DECLARE

BEGIN

```
INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('1', 'Ploiesti', 'Centru');  
INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('2', 'Ploiesti', 'Nord');  
INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('3', 'Bucuresti', 'Romana');  
INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('4', 'Bucuresti', 'Universitate');  
INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('5', 'Bucuresti', 'Victoriei');  
INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('6', 'Cluj-Napoca', 'Centru');  
INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('7', 'Sibiu', 'Cisnadiiei');  
INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('8', 'Brasov', 'Centru');  
INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('9', 'Arad', 'Centru');  
INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('10', 'Iasi', 'Centru');
```

COMMIT;

END;

/

The screenshot shows a database management tool interface. On the left, a tree view displays the database structure, including tables like COMENZI, DATE\_PERSONE, and LOCATII. The main window shows the execution of an SQL script. The script contains a series of INSERT statements for the Locatii table, followed by a COMMIT and a SELECT statement. The results of the SELECT statement are displayed in a table at the bottom of the window.

```
70  
71 DECLARE  
72 BEGIN  
73 INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('1', 'Ploiesti', 'Centru');  
74 INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('2', 'Ploiesti', 'Nord');  
75 INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('3', 'Bucuresti', 'Romana');  
76 INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('4', 'Bucuresti', 'Universitate');  
77 INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('5', 'Bucuresti', 'Victoriei');  
78 INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('6', 'Cluj-Napoca', 'Centru');  
79 INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('7', 'Sibiu', 'Cisnadiiei');  
80 INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('8', 'Brasov', 'Centru');  
81 INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('9', 'Arad', 'Centru');  
82 INSERT INTO Locatii (ID_locatie, oras, zona) VALUES ('10', 'Iasi', 'Centru');  
83  
84 COMMIT;  
85  
86 /  
87  
88 select * from locatii;  
89
```

Script Output x Query Result x  
All Rows Fetched: 10 in 0,016 seconds

ID_LOCATIE	ORAS	ZONA
1 1	Ploiesti	Centru
2 2	Ploiesti	Nord
3 3	Bucuresti	Romana
4 4	Bucuresti	Universitate
5 5	Bucuresti	Victoriei
6 6	Cluj-Napoca	Centru
7 7	Sibiu	Cisnadiiei
8 8	Brasov	Centru
9 9	Arad	Centru
10 10	Iasi	Centru

### 3. Tabela Produse

DECLARE

BEGIN

INSERT INTO Produse (ID\_produs, denumire, pret) VALUES ('1', 'Iced Caramel Macchiato', 17);

INSERT INTO Produse (ID\_produs, denumire, pret) VALUES ('2', 'Cappuccino', 13);

INSERT INTO Produse (ID\_produs, denumire, pret) VALUES ('3', 'Flat White', 10);

INSERT INTO Produse (ID\_produs, denumire, pret) VALUES ('4', 'Matcha Green Tea Latte', 20);

INSERT INTO Produse (ID\_produs, denumire, pret) VALUES ('5', 'Ceai de hibiscus', 12);

INSERT INTO Produse (ID\_produs, denumire, pret) VALUES ('6', 'Gingerbread Latte', 20);

INSERT INTO Produse (ID\_produs, denumire, pret) VALUES ('7', 'Ciocolata calda', 11);

INSERT INTO Produse (ID\_produs, denumire, pret) VALUES ('8', 'Croissant cu zmeura', 7);

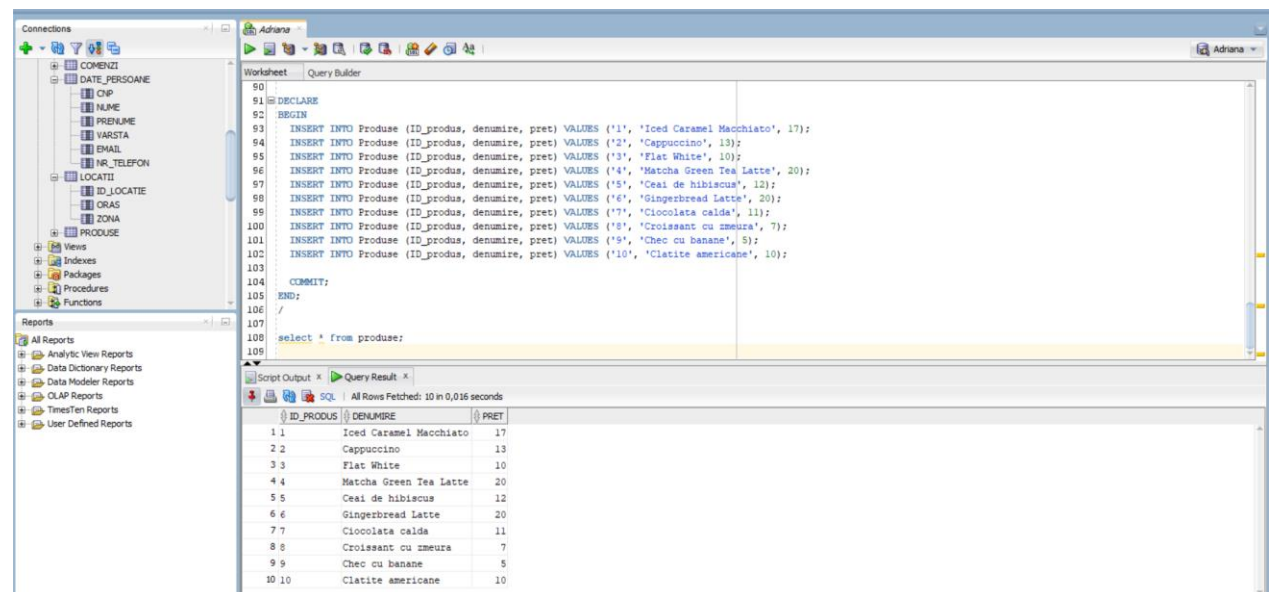
INSERT INTO Produse (ID\_produs, denumire, pret) VALUES ('9', 'Chec cu banane', 5);

INSERT INTO Produse (ID\_produs, denumire, pret) VALUES ('10', 'Clatite americane', 10);

COMMIT;

END;

/



The screenshot displays the SQL Developer interface. The 'Worksheet' tab contains the following SQL script:

```
90  
91 DECLARE  
92 BEGIN  
93 INSERT INTO Produse (ID_produs, denumire, pret) VALUES ('1', 'Iced Caramel Macchiato', 17);  
94 INSERT INTO Produse (ID_produs, denumire, pret) VALUES ('2', 'Cappuccino', 13);  
95 INSERT INTO Produse (ID_produs, denumire, pret) VALUES ('3', 'Flat White', 10);  
96 INSERT INTO Produse (ID_produs, denumire, pret) VALUES ('4', 'Matcha Green Tea Latte', 20);  
97 INSERT INTO Produse (ID_produs, denumire, pret) VALUES ('5', 'Ceai de hibiscus', 12);  
98 INSERT INTO Produse (ID_produs, denumire, pret) VALUES ('6', 'Gingerbread Latte', 20);  
99 INSERT INTO Produse (ID_produs, denumire, pret) VALUES ('7', 'Ciocolata calda', 11);  
100 INSERT INTO Produse (ID_produs, denumire, pret) VALUES ('8', 'Croissant cu zmeura', 7);  
101 INSERT INTO Produse (ID_produs, denumire, pret) VALUES ('9', 'Chec cu banane', 5);  
102 INSERT INTO Produse (ID_produs, denumire, pret) VALUES ('10', 'Clatite americane', 10);  
103  
104 COMMIT;  
105 END;  
106 /  
107  
108 select * from produse;  
109
```

The 'Query Result' tab shows the output of the query, displaying 10 rows of data:

ID_PRODUS	DENUMIRE	PRET
1 1	Iced Caramel Macchiato	17
2 2	Cappuccino	13
3 3	Flat White	10
4 4	Matcha Green Tea Latte	20
5 5	Ceai de hibiscus	12
6 6	Gingerbread Latte	20
7 7	Ciocolata calda	11
8 8	Croissant cu zmeura	7
9 9	Chec cu banane	5
10 10	Clatite americane	10

## 4. Tabela Comenzi

BEGIN

```
INSERT INTO Comenzi VALUES ('1','Online',TO_DATE('02-JAN-2023', 'DD-MON-YYYY'),  
'livrata', 12, '5');
```

```
INSERT INTO Comenzi VALUES ('2','In magazin', TO_DATE('03-JAN-2023', 'DD-MON-  
YYYY'), 'plasata', 22, '7');
```

```
INSERT INTO Comenzi VALUES ('3','Online', TO_DATE('02-FEB-2023', 'DD-MON-YYYY'),  
'plasata', 14, '8');
```

```
INSERT INTO Comenzi VALUES ('4','Online', TO_DATE('22-DEC-2022', 'DD-MON-  
YYYY'), 'livrata', 40, '6');
```

```
INSERT INTO Comenzi VALUES ('5','In magazin', TO_DATE('28-DEC-2022', 'DD-MON-  
YYYY'), 'ridicata', 30, '3');
```

```
INSERT INTO Comenzi VALUES ('6','In magazin', TO_DATE('30-DEC-2022', 'DD-MON-  
YYYY'), 'plasata', 10, '10');
```

```
INSERT INTO Comenzi VALUES ('7','In magazin', TO_DATE('05-JAN-2023', 'DD-MON-  
YYYY'), 'pregatita', 15, '9');
```

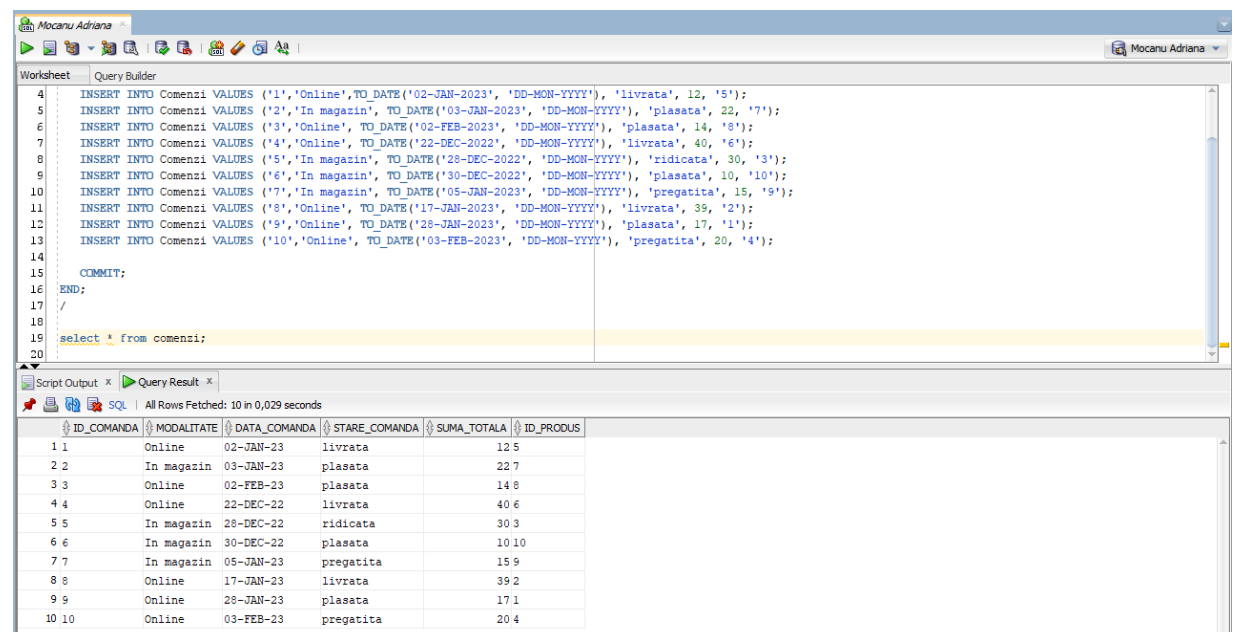
```
INSERT INTO Comenzi VALUES ('8','Online', TO_DATE('17-JAN-2023', 'DD-MON-YYYY'),  
'livrata', 39, '2');
```

```
INSERT INTO Comenzi VALUES ('9','Online', TO_DATE('28-JAN-2023', 'DD-MON-YYYY'),  
'plasata', 17, '1');
```

```
INSERT INTO Comenzi VALUES ('10','Online', TO_DATE('03-FEB-2023', 'DD-MON-  
YYYY'), 'pregatita', 20, '4');
```

```
COMMIT;  
END;
```

/



The screenshot shows a database query tool interface. The top part displays a list of SQL commands being executed, including 10 INSERT statements and a COMMIT. The bottom part shows the query results in a table format.

ID_COMANDA	MODALITATE	DATA_COMANDA	STARE_COMANDA	SUMA_TOTALA	ID_PRODUS
1	Online	02-JAN-23	livrata	12.5	
2	In magazin	03-JAN-23	plasata	22.7	
3	Online	02-FEB-23	plasata	14.8	
4	Online	22-DEC-22	livrata	40.6	
5	In magazin	28-DEC-22	ridicata	30.3	
6	In magazin	30-DEC-22	plasata	10.10	
7	In magazin	05-JAN-23	pregatita	15.9	
8	Online	17-JAN-23	livrata	39.2	
9	Online	28-JAN-23	plasata	17.1	
10	Online	03-FEB-23	pregatita	20.4	

## 5. Tabela Angajati

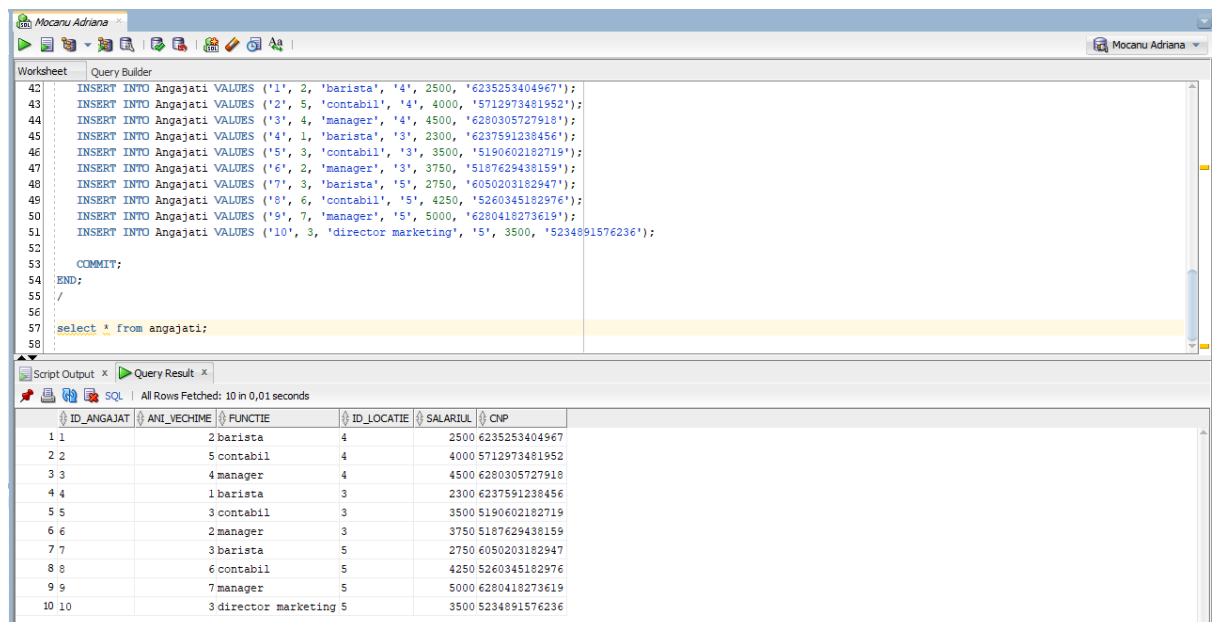
BEGIN

```
INSERT INTO Angajati VALUES ('1', 2, 'barista', '4', 2500, '6235253404967');
INSERT INTO Angajati VALUES ('2', 5, 'contabil', '4', 4000, '5712973481952');
INSERT INTO Angajati VALUES ('3', 4, 'manager', '4', 4500, '6280305727918');
INSERT INTO Angajati VALUES ('4', 1, 'barista', '3', 2300, '6237591238456');
INSERT INTO Angajati VALUES ('5', 3, 'contabil', '3', 3500, '5190602182719');
INSERT INTO Angajati VALUES ('6', 2, 'manager', '3', 3750, '5187629438159');
INSERT INTO Angajati VALUES ('7', 3, 'barista', '5', 2750, '6050203182947');
INSERT INTO Angajati VALUES ('8', 6, 'contabil', '5', 4250, '5260345182976');
INSERT INTO Angajati VALUES ('9', 7, 'manager', '5', 5000, '6280418273619');
INSERT INTO Angajati VALUES ('10', 3, 'director marketing', '5', 3500, '5234891576236');
```

COMMIT;

END;

/



The screenshot shows a SQL query editor window with a 'Query Builder' tab. The query contains ten INSERT statements for the 'Angajati' table, followed by a COMMIT, an END statement, and a SELECT statement to verify the data. Below the query, the 'Query Result' tab displays the data fetched from the database.

ID_ANGAJAT	ANUL_VECHIME	FUNCTIE	ID_LOCATIE	SALARIU	CNP
1	1	2 barista	4	2500	6235253404967
2	2	5 contabil	4	4000	5712973481952
3	3	4 manager	4	4500	6280305727918
4	4	1 barista	3	2300	6237591238456
5	5	3 contabil	3	3500	5190602182719
6	6	2 manager	3	3750	5187629438159
7	7	3 barista	5	2750	6050203182947
8	8	6 contabil	5	4250	5260345182976
9	9	7 manager	5	5000	6280418273619
10	10	3 director marketing	5	3500	5234891576236

## 6. Tabela clienti

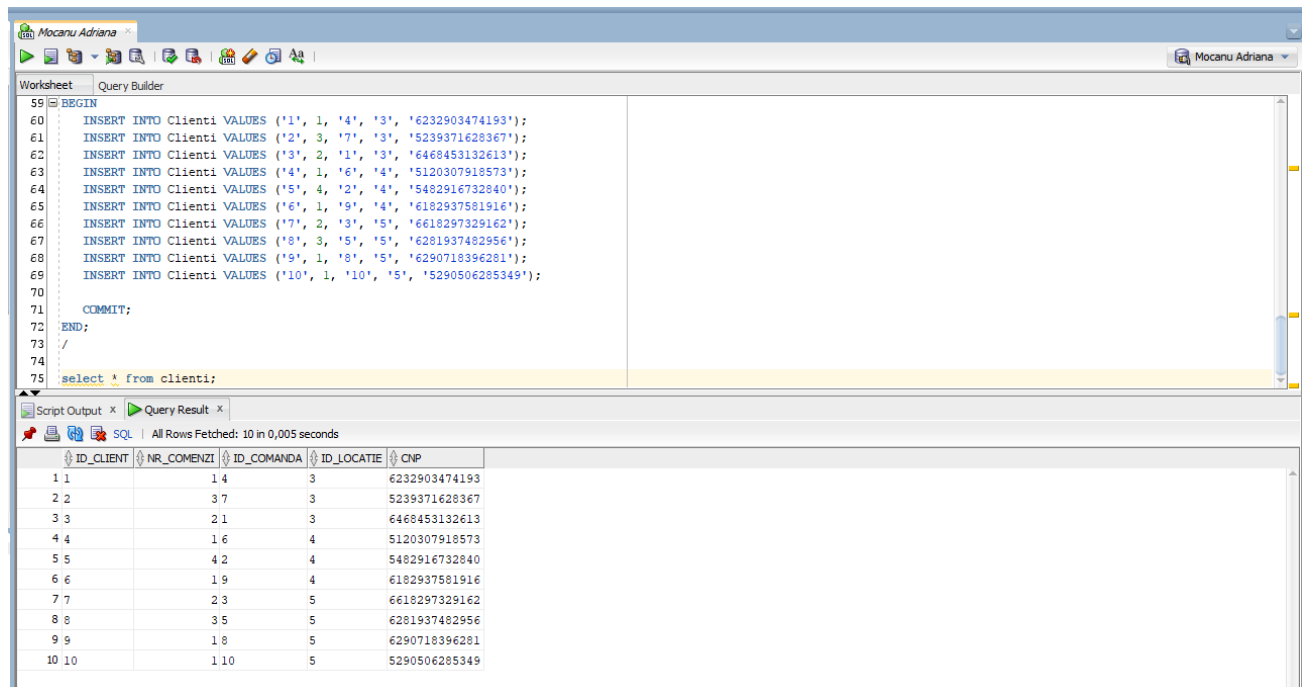
BEGIN

```
INSERT INTO Clienti VALUES ('1', 1, '4', '3', '6232903474193');
INSERT INTO Clienti VALUES ('2', 3, '7', '3', '5239371628367');
INSERT INTO Clienti VALUES ('3', 2, '1', '3', '6468453132613');
INSERT INTO Clienti VALUES ('4', 1, '6', '4', '5120307918573');
INSERT INTO Clienti VALUES ('5', 4, '2', '4', '5482916732840');
INSERT INTO Clienti VALUES ('6', 1, '9', '4', '6182937581916');
INSERT INTO Clienti VALUES ('7', 2, '3', '5', '6618297329162');
INSERT INTO Clienti VALUES ('8', 3, '5', '5', '6281937482956');
INSERT INTO Clienti VALUES ('9', 1, '8', '5', '6290718396281');
INSERT INTO Clienti VALUES ('10', 1, '10', '5', '5290506285349');
```

COMMIT;

END;

/



The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and help. The main window is divided into a 'Worksheet' and a 'Query Builder' tab. The 'Worksheet' tab displays the following SQL script:

```
59 BEGIN
60 INSERT INTO Clienti VALUES ('1', 1, '4', '3', '6232903474193');
61 INSERT INTO Clienti VALUES ('2', 3, '7', '3', '5239371628367');
62 INSERT INTO Clienti VALUES ('3', 2, '1', '3', '6468453132613');
63 INSERT INTO Clienti VALUES ('4', 1, '6', '4', '5120307918573');
64 INSERT INTO Clienti VALUES ('5', 4, '2', '4', '5482916732840');
65 INSERT INTO Clienti VALUES ('6', 1, '9', '4', '6182937581916');
66 INSERT INTO Clienti VALUES ('7', 2, '3', '5', '6618297329162');
67 INSERT INTO Clienti VALUES ('8', 3, '5', '5', '6281937482956');
68 INSERT INTO Clienti VALUES ('9', 1, '8', '5', '6290718396281');
69 INSERT INTO Clienti VALUES ('10', 1, '10', '5', '5290506285349');
70
71 COMMIT;
72 END;
73 /
74
75 select * from clienti;
```

Below the script, the 'Script Output' and 'Query Result' tabs are visible. The 'Query Result' tab shows the results of the SQL script, displaying 10 rows of data. The columns are labeled: ID\_CLIENT, NR\_COMENZI, ID\_COMANDA, ID\_LOCATIE, and CNP. The data is as follows:

ID_CLIENT	NR_COMENZI	ID_COMANDA	ID_LOCATIE	CNP
1 1	1 4	3		6232903474193
2 2	3 7	3		5239371628367
3 3	2 1	3		6468453132613
4 4	1 6	4		5120307918573
5 5	4 2	4		5482916732840
6 6	1 9	4		6182937581916
7 7	2 3	5		6618297329162
8 8	3 5	5		6281937482956
9 9	1 8	5		6290718396281
10 10	1 10	5		5290506285349

## B. Interacțiunea cu serverul Oracle prin intermediul comenzilor SQL

### (LDD și LMD)

1. Să se majoreze prețul produselor al căror preț actual este mai mic de 20 lei cu 20%.

declare

procent number:=0.2;

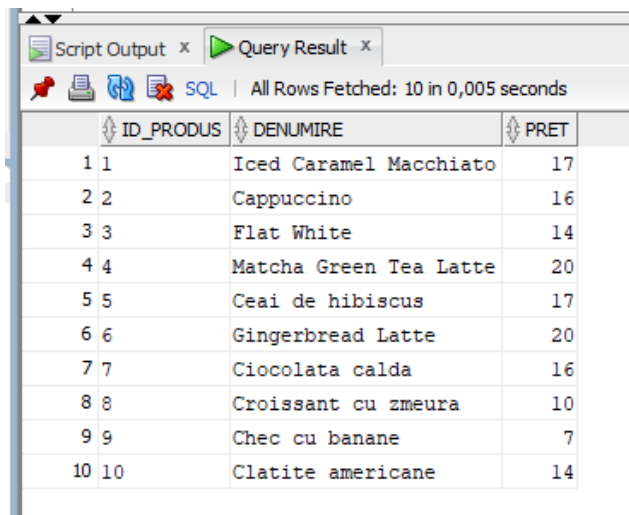
begin

update produse

set pret=(pret\*procent)+pret where pret<20;

end;

/



ID_PRODUS	DENUMIRE	PRET
1 1	Iced Caramel Macchiato	17
2 2	Cappuccino	16
3 3	Flat White	14
4 4	Matcha Green Tea Latte	20
5 5	Ceai de hibiscus	17
6 6	Gingerbread Latte	20
7 7	Ciocolata calda	16
8 8	Croissant cu zmeura	10
9 9	Chec cu banane	7
10 10	Clatite americane	14



3	declare	
4	procent number:=0.2;	
5	begin	
6	update produse	
7	set pret=(pret*procent)+pret where pret<20;	
8	end;	
9	/	
10		

Script Output x Query Result x		
SQL   All Rows Fetched: 10 in 0,006 seconds		
ID_PRODUS	DENUMIRE	PRET
1 1	Iced Caramel Macchiato	20
2 2	Cappuccino	19
3 3	Flat White	17
4 4	Matcha Green Tea Latte	20
5 5	Ceai de hibiscus	20
6 6	Gingerbread Latte	20
7 7	Ciocolata calda	19
8 8	Croissant cu zmeura	12
9 9	Chec cu banane	8
10 10	Clatite americane	17

2. Să se afișeze denumirile produselor din comenzile care au id număr par.

BEGIN

FOR rec IN (

SELECT p.denumire

FROM Comenzi c

INNER JOIN Produse p ON c.ID\_produs = p.ID\_produs

WHERE MOD(c.ID\_comanda, 2) = 0

)

LOOP

DBMS\_OUTPUT.PUT\_LINE(rec.denumire);

END LOOP;

END;

/

```

14 BEGIN
15   FOR rec IN (
16     SELECT p.denumire
17     FROM Comenzi c
18     INNER JOIN Produse p ON c.ID_produc = p.ID_produc
19     WHERE MOD(c.ID_comanda, 2) = 0
20   )
21   LOOP
22     DBMS_OUTPUT.PUT_LINE(rec.denumire);
23   END LOOP;
24 END;
25 /
26

```

Script Output x Query Result x

Task completed in 0,032 seconds

Ciocolata calda  
Gingerbread Latte  
Clatite americane  
Cappuccino  
Matcha Green Tea Latte

PL/SQL procedure successfully completed.

- Să se modifice toate denumirile funcțiilor astfel încât acestea să fie scrise cu litere mari.

BEGIN

UPDATE angajati

SET functie=UPPER(functie);

END;

/

```

27 BEGIN
28   UPDATE angajati
29   SET functie=UPPER(functie);
30 END;
31 /
32
33 select * from angajati;

```

Script Output x Query Result x

All Rows Fetched: 10 in 0,004 seconds

ID_ANGAJAT	ANI_VECHIME	FUNCTIE	ID_LOCATIE	SALARIUL	CNP
1 1		2 BARISTA	4	2500	6235253404967
2 2		5 CONTABIL	4	4000	5712973481952
3 3		4 MANAGER	4	4500	6280305727918
4 4		1 BARISTA	3	2300	6237591238456
5 5		3 CONTABIL	3	3500	5190602182719
6 6		2 MANAGER	3	3750	5187629438159
7 7		3 BARISTA	5	2750	6050203182947
8 8		6 CONTABIL	5	4250	5260345182976
9 9		7 MANAGER	5	5000	6280418273619
10 10		3 DIRECTOR MARKETING	5	3500	5234891576236

- Să se afișeze numele, prenumele și e-mailul clientului al cărui CNP începe cu cifra 5.

BEGIN

FOR rec IN (

SELECT dp.num, dp.prenume, dp.email

```

FROM Clienti c

INNER JOIN Date_persoane dp ON c.CNP = dp.CNP

WHERE SUBSTR(c.CNP, 1, 1) = '5'

)

LOOP

    DBMS_OUTPUT.PUT_LINE('Nume: ' || rec.num);

    DBMS_OUTPUT.PUT_LINE('Prenume: ' || rec.prenume);

    DBMS_OUTPUT.PUT_LINE('E-mail: ' || rec.email);

END LOOP;

END;

/

```

The screenshot shows the Oracle SQL Developer interface. The top pane displays a PL/SQL script with line numbers 35 to 49. The script defines a loop that iterates over records from the 'Clienti' table, joining with 'Date\_persoane' where the first digit of the CNP is '5'. It then uses DBMS\_OUTPUT.PUT\_LINE to display the name, prename, and email of each record. The bottom pane shows the 'Script Output' window, which displays the results of the script execution: four records with their respective names, prenames, and emails. The status bar at the bottom indicates 'Task completed in 0,044 seconds' and 'PL/SQL procedure successfully completed.'

```

35 BEGIN
36   FOR rec IN (
37     SELECT dp.num, dp.prenume, dp.email
38     FROM Clienti c
39     INNER JOIN Date_persoane dp ON c.CNP = dp.CNP
40     WHERE SUBSTR(c.CNP, 1, 1) = '5'
41   )
42   LOOP
43     DBMS_OUTPUT.PUT_LINE('Nume: ' || rec.num);
44     DBMS_OUTPUT.PUT_LINE('Prenume: ' || rec.prenume);
45     DBMS_OUTPUT.PUT_LINE('E-mail: ' || rec.email);
46   END LOOP;
47 END;
48 /
49

```

Script Output x Query Result x

Task completed in 0,044 seconds

```

Prenume: Valentin
E-mail: tvalentin@gmail.com
Nume: Moise
Prenume: Alexandra
E-mail: malexandra@gmail.com
Nume: Neacsu
Prenume: Andrei
E-mail: nandrei@yahoo.com

```

PL/SQL procedure successfully completed.

5. Să se afișeze prenumele, vârsta și numărul de telefon al clienților care au comenzi într-un anumit oraș.

```

DECLARE

v_oras VARCHAR2(20) := '&introduceti_orasul';

BEGIN

FOR rec IN (

    SELECT dp.prenume, dp.varsta, dp.nr_telefon

    FROM clienti c

    JOIN date_persoane dp ON c.CNP = dp.CNP

    JOIN locatii l ON c.ID_locatie = l.ID_locatie

    WHERE l.oras = v_oras

)

LOOP

```

```

DBMS_OUTPUT.PUT_LINE('Prenume: ' || rec.prenume);

DBMS_OUTPUT.PUT_LINE('Vârsta: ' || rec.varsta);

DBMS_OUTPUT.PUT_LINE('Număr telefon: ' || rec.nr_telefon);

END LOOP;

```

```
END;
```

```
/
```

```

50 DECLARE
51     v_oras VARCHAR2(20) := 'introduceti_orasul';
52 BEGIN
53     FOR rec IN (
54         SELECT dp.prenume, dp.varsta, dp.nr_telefon...
55         FROM clienti c
56         JOIN date_persoane dp ON c.CNP = dp.CNP
57         JOIN locatii l ON c.ID_locatie = l.ID_locatie
58         WHERE l.oras = v_oras
59     )
60     LOOP
61         DBMS_OUTPUT.PUT_LINE('Prenume: ' || rec.prenume);
62         DBMS_OUTPUT.PUT_LINE('Vârsta: ' || rec.varsta);
63         DBMS_OUTPUT.PUT_LINE('Număr telefon: ' || rec.nr_telefon);
64     END LOOP;
65 END;
66 /
67

```

Script Output x Query Result x

Task completed in 17,946 seconds

```

Vârsta: 24
Numar telefon: +40736529401
Prenume: Alin
Vârsta: 40
Numar telefon: +40718492653
Prenume: Andrei
Vârsta: 18
Numar telefon: +40739182746

```

## C. Structuri alternative / repetitive

1. Comparăți prețul produselor având id-ul citit de la tastatură, prețul fix de 15 de ron și afișați dacă produsul este ieftin, scump sau normal.

```
declare
```

```
pret_fix number(5):=15;
```

```
pret_p produse.pret%type;
```

```
begin
```

```
select pret into pret_p from produse where id_produs=&p;
```

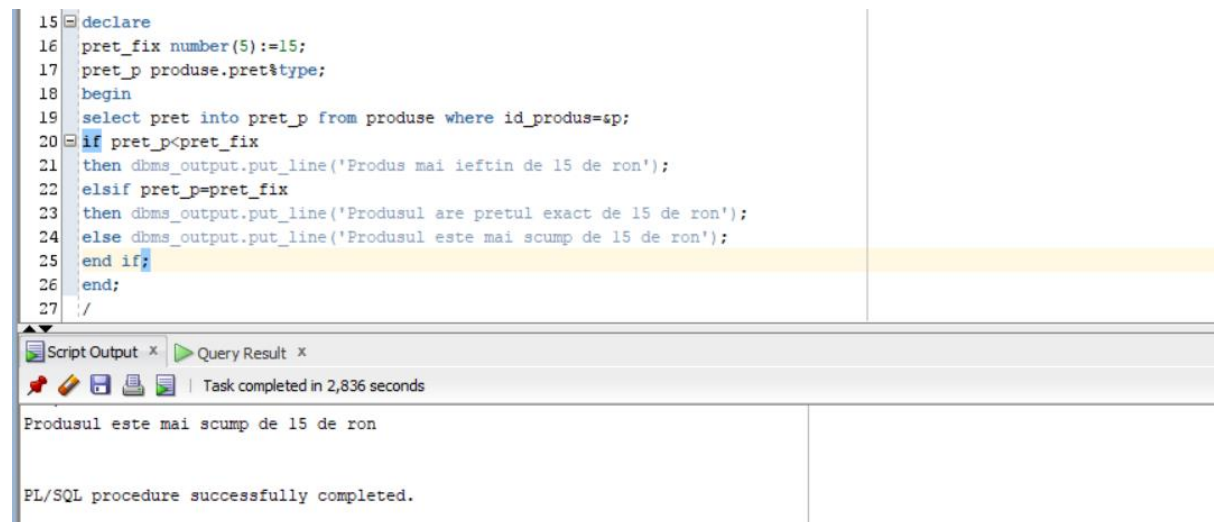
```
if pret_p<pret_fix
```

```
then dbms_output.put_line('Produs mai ieftin de 15 de ron');
```

```

elsif pret_p=pret_fix
then dbms_output.put_line('Produsul are pretul exact de 15 de ron');
else dbms_output.put_line('Produsul este mai scump de 15 de ron');
end if;
end;
/

```



The screenshot shows the Oracle SQL Developer interface. The top pane displays a PL/SQL script with the following code:

```

15 declare
16   pret_fix number(5):=15;
17   pret_p produse.pret%type;
18 begin
19   select pret into pret_p from produse where id_produs=sp;
20   if pret_p<pret_fix
21   then dbms_output.put_line('Produs mai ieftin de 15 de ron');
22   elsif pret_p=pret_fix
23   then dbms_output.put_line('Produsul are pretul exact de 15 de ron');
24   else dbms_output.put_line('Produsul este mai scump de 15 de ron');
25   end if;
26 end;
27 /

```

The bottom pane shows the 'Script Output' tab with the following message:

```

Produsul este mai scump de 15 de ron

```

Below the output, a status bar indicates: 'Task completed in 2,836 seconds' and 'PL/SQL procedure successfully completed.'

2. Să se afișeze în ordine produsele cu id-ul între 5 și 10, atât timp cât prețul acestora este mai mare decât prețul mediu.

```

declare
pret_p produse.pret%type;
pret_mediu pret_p%type;
i number(4):=5;
begin
loop
select avg(pret) into pret_mediu from produse;

select pret into pret_p from produse where id_produs=i;
if pret_p>pret_mediu and i<10
then dbms_output.put_line('Produsul cu id-ul '||i||' are pretul: '||pret_p);
end if;
i:=i+1;
exit when i>9;
end loop;
end;

```

```

/
29 declare
30 pret_p produse.pret%type;
31 pret_mediu pret_p%type;
32 i number(4):=5;
33 begin
34 loop
35 select avg(pret) into pret_mediu from produse;
36
37 select pret into pret_p from produse where id_produc=i;
38 if pret_p>pret_mediu and i<10
39 then dbms_output.put_line('Produsul cu id-ul '||i||' are pretul: '||pret_p);
40 end if;
41 i:=i+1;
42 exit when i>9;
43 end loop;
44 end;
45

```

Script Output x Query Result x

Task completed in 0,046 seconds

```

Produsul cu id-ul 5 are pretul: 20
Produsul cu id-ul 6 are pretul: 20
Produsul cu id-ul 7 are pretul: 19

```

## D. Tratarea excepțiilor

1. Creați un bloc PL/SQL prin care să creșteți cu 5% salariul angajaților cu funcția primită de la tastatură. Afișați numărul de înregistrări modificate. Ridicați o excepție în cazul în care nu sunt modificări.

declare

nume\_functie angajati.functie%type:='&f';

e exception;

begin

update angajati set salariul=1.05\*salariul where id\_angajat in(select id\_angajat from angajati  
where functie=nume\_functie);

if sql%rowcount=0

then raise e;

else dbms\_output.put\_line('Numarul de modificari este '||sql%rowcount);

end if;

exception

when e

then dbms\_output.put\_line('Nu s-a facut nicio modificare');

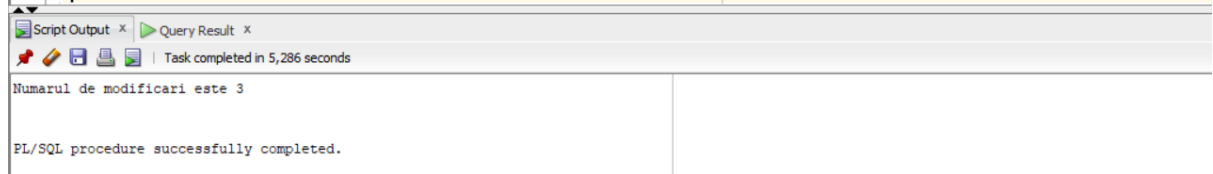
end;

/

```

47 declare
48     nume_functie angajati.functie%type:= 'sf';
49     e exception;
50 begin
51     update angajati set salariul=1.05*salariul where id_angajat in(select id_angajat from angajati
52     where functie=nume_functie);
53     if sql%rowcount=0
54     then raise e;
55     else dbms_output.put_line('Numarul de modificari este '||sql%rowcount);
56     end if;
57     exception
58     when e
59     then dbms_output.put_line('Nu s-a facut nicio modificare');
60 end;
61 /
62

```



2. Să se afișeze angajatul cu codul 102. Să se trateze eroarea apărută în cazul în care nu există niciun angajat cu acest cod.

DECLARE

v\_id\_angajat VARCHAR2(3) := '102';

v\_ani\_vechime NUMBER(2);

v\_functie VARCHAR2(30);

v\_id\_locatie VARCHAR2(20);

v\_salariul NUMBER(6);

v\_cnp CHAR(13);

BEGIN

SELECT ani\_vechime, functie, ID\_locatie, salariul, CNP

INTO v\_ani\_vechime, v\_functie, v\_id\_locatie, v\_salariul, v\_cnp

FROM Angajati

WHERE ID\_angajat = v\_id\_angajat;

DBMS\_OUTPUT.PUT\_LINE('Angajatul cu codul ' || v\_id\_angajat || ' exista:');

DBMS\_OUTPUT.PUT\_LINE('Ani vechime: ' || v\_ani\_vechime);

DBMS\_OUTPUT.PUT\_LINE('Functie: ' || v\_functie);

DBMS\_OUTPUT.PUT\_LINE('ID locatie: ' || v\_id\_locatie);

DBMS\_OUTPUT.PUT\_LINE('Salariu: ' || v\_salariul);

DBMS\_OUTPUT.PUT\_LINE('CNP: ' || v\_cnp);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista angajat cu codul ' || v\_id\_angajat);

END;

```
/
66 DECLARE
67   v_id_angajat VARCHAR2(3) := '102';
68   v_an_i_vehime NUMBER(2);
69   v_functie VARCHAR2(30);
70   v_id_locatie VARCHAR2(20);
71   v_salariul NUMBER(6);
72   v_cnp CHAR(13);
73 BEGIN
74   SELECT ani_vehime, functie, ID_locatie, salariul, CNP
75   INTO v_an_i_vehime, v_functie, v_id_locatie, v_salariul, v_cnp
76   FROM Angajati
77   WHERE ID_angajat = v_id_angajat;
78
79   DBMS_OUTPUT.PUT_LINE('Angajatul cu codul ' || v_id_angajat || ' exista:');
80   DBMS_OUTPUT.PUT_LINE('Ani vehime: ' || v_an_i_vehime);
81   DBMS_OUTPUT.PUT_LINE('Functie: ' || v_functie);
82   DBMS_OUTPUT.PUT_LINE('ID locatie: ' || v_id_locatie);
83   DBMS_OUTPUT.PUT_LINE('Salariu: ' || v_salariul);
84   DBMS_OUTPUT.PUT_LINE('CNP: ' || v_cnp);
85 EXCEPTION
86   WHEN NO_DATA_FOUND THEN
87     DBMS_OUTPUT.PUT_LINE('Nu exista angajat cu codul ' || v_id_angajat);
88 END;
89 /
90
91
```

Script Output x Query Result x

Task completed in 0,04 seconds

Nu exista angajat cu codul 102

PL/SQL procedure successfully completed.

3. Creați un bloc PL/SQL prin care să se afișeze prenumele și salariul pentru angajații cu funcția introdusă de la tastatură, tratând și excepțiile implicite care pot apărea.

DECLARE

v\_functie VARCHAR2(30) := '&functia';

v\_prenume Date\_persoane.prenume%TYPE;

v\_salariu Angajati.salariul%TYPE;

BEGIN

SELECT dp.prenume, a.salariul

INTO v\_prenume, v\_salariu

FROM Angajati a

INNER JOIN Date\_persoane dp ON a.CNP = dp.CNP

WHERE a.functie = v\_functie;

DBMS\_OUTPUT.PUT\_LINE('Angajatii cu functia ' || v\_functie || ':');

DBMS\_OUTPUT.PUT\_LINE('Prenume: ' || v\_prenume);

DBMS\_OUTPUT.PUT\_LINE('Salariu: ' || v\_salariu);

EXCEPTION



WHEN NO\_DATA\_FOUND THEN

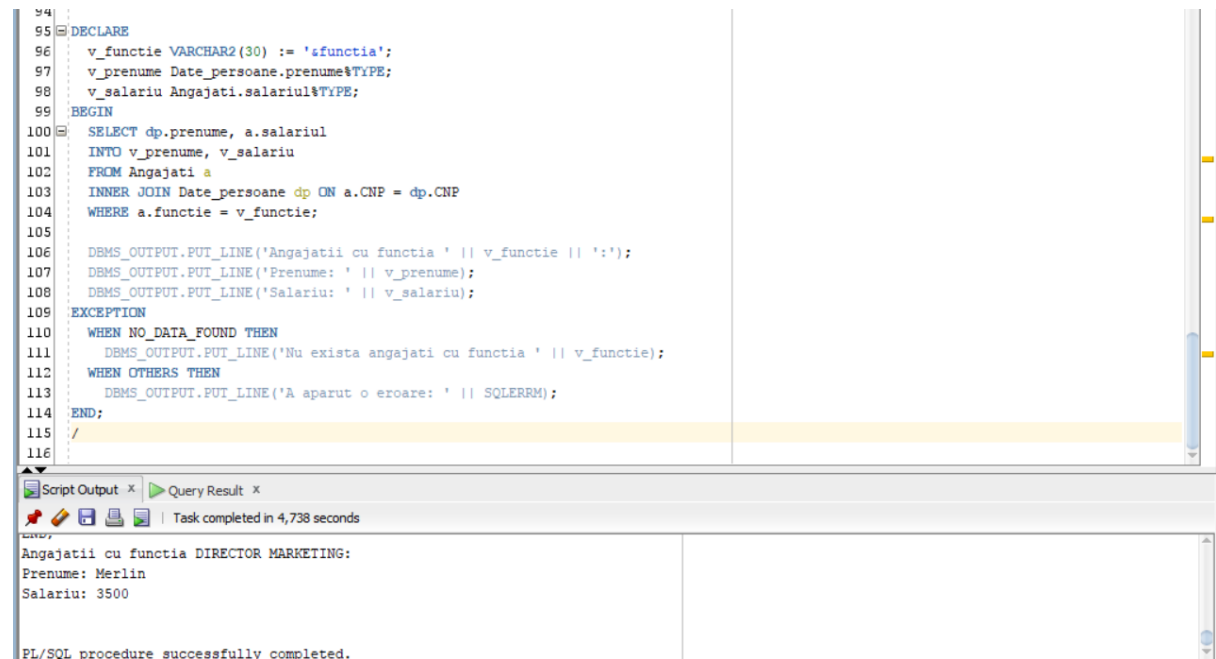
DBMS\_OUTPUT.PUT\_LINE('Nu exista angajati cu functia ' || v\_functie);

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare: ' || SQLERRM);

END;

/



The screenshot shows a PL/SQL procedure in a script editor. The procedure declares variables `v_functie`, `v_prenume`, and `v_salariu`. It uses a `SELECT` statement with an `INNER JOIN` to retrieve data from `Angajati` and `Date_persoane` tables. The procedure includes `DBMS_OUTPUT.PUT_LINE` statements to display the results. It also has an `EXCEPTION` block to handle `NO_DATA_FOUND` and other errors. Below the script, the 'Script Output' window shows the execution results, indicating that the task was completed in 4,738 seconds and displaying the output of the procedure.

```
94  
95 DECLARE  
96   v_functie VARCHAR2(30) := 'sfunctia';  
97   v_prenume Date_persoane.prenume%TYPE;  
98   v_salariu Angajati.salariul%TYPE;  
99 BEGIN  
100   SELECT dp.prenume, a.salariul  
101     INTO v_prenume, v_salariu  
102     FROM Angajati a  
103     INNER JOIN Date_persoane dp ON a.CNP = dp.CNP  
104     WHERE a.functie = v_functie;  
105  
106   DBMS_OUTPUT.PUT_LINE('Angajatii cu functia ' || v_functie || ':');  
107   DBMS_OUTPUT.PUT_LINE('Prenume: ' || v_prenume);  
108   DBMS_OUTPUT.PUT_LINE('Salariu: ' || v_salariu);  
109 EXCEPTION  
110   WHEN NO_DATA_FOUND THEN  
111     DBMS_OUTPUT.PUT_LINE('Nu exista angajati cu functia ' || v_functie);  
112   WHEN OTHERS THEN  
113     DBMS_OUTPUT.PUT_LINE('A aparut o eroare: ' || SQLERRM);  
114 END;  
115 /  
116
```

Script Output x Query Result x  
Task completed in 4,738 seconds  
Angajatii cu functia DIRECTOR MARKETING:  
Prenume: Merlin  
Salariu: 3500  
PL/SQL procedure successfully completed.

4. Să se afișeze toți angajații cu mai puțin de 5 ani vechime. În cazul în care nu există angajați, să se arunce o excepție explicită.

DECLARE

v\_limit NUMBER(2) := 5;

e exception;

BEGIN

IF SQL%NOTFOUND

THEN RAISE e;

else

FOR angajat IN (SELECT \* FROM Angajati WHERE ani\_vechime < v\_limit) LOOP

DBMS\_OUTPUT.PUT\_LINE('ID\_angajat: ' || angajat.ID\_angajat);

DBMS\_OUTPUT.PUT\_LINE('Ani vechime: ' || angajat.ani\_vechime);

DBMS\_OUTPUT.PUT\_LINE('Functie: ' || angajat.functie);

DBMS\_OUTPUT.PUT\_LINE('ID\_locatie: ' || angajat.ID\_locatie);

DBMS\_OUTPUT.PUT\_LINE('Salariu: ' || angajat.salariul);

```

        DBMS_OUTPUT.PUT_LINE('CNP: ' || angajat.CNP);

    END LOOP;

    END IF;

EXCEPTION

    WHEN e

    THEN

        dbms_output.put_line('Nu există angajați cu mai puțin de 5 ani vechime');

    END;

/

```

The screenshot shows the Oracle SQL Developer interface. The top pane displays a PL/SQL script with line numbers 120 to 140. The script includes a loop to process employees and an exception handler. The bottom pane shows the 'Script Output' tab with the results of the script execution, indicating that the task was completed in 0.037 seconds. The output lists the details for two employees: one with CNP 5187629438159 and another with CNP 6050203182947.

```

120 e exception;
121 BEGIN
122 IF SQL%NOTFOUND
123 THEN RAISE e;
124 else
125 FOR angajat IN (SELECT * FROM Angajati WHERE ani_vechime < v_limit) LOOP
126   DBMS_OUTPUT.PUT_LINE('ID_angajat: ' || angajat.ID_angajat);
127   DBMS_OUTPUT.PUT_LINE('Ani vechime: ' || angajat.ani_vechime);
128   DBMS_OUTPUT.PUT_LINE('Functie: ' || angajat.functie);
129   DBMS_OUTPUT.PUT_LINE('ID_locatie: ' || angajat.ID_locatie);
130   DBMS_OUTPUT.PUT_LINE('Salariu: ' || angajat.salariul);
131   DBMS_OUTPUT.PUT_LINE('CNP: ' || angajat.CNP);
132 END LOOP;
133 END IF;
134 EXCEPTION
135 WHEN e
136 THEN
137   dbms_output.put_line('Nu există angajați cu mai puțin de 5 ani vechime');
138 END;
139 /
140

```

Script Output x Query Result x

Task completed in 0,037 seconds

```

CNP: 5187629438159
ID_angajat: 7
Ani vechime: 3
Functie: BARISTA
ID_locatie: 5
Salariu: 2888
CNP: 6050203182947
ID_angajat: 10
Ani vechime: 3
Functie: DIRECTOR MARKETING
ID_locatie: 5
Salariu: 3500

```

## E. Gestionarea cursorilor

### Cursor explicit

1. Să se selecteze toate comenzile realizate online și să se calculeze valoarea totală a acestora.

```
DECLARE
```

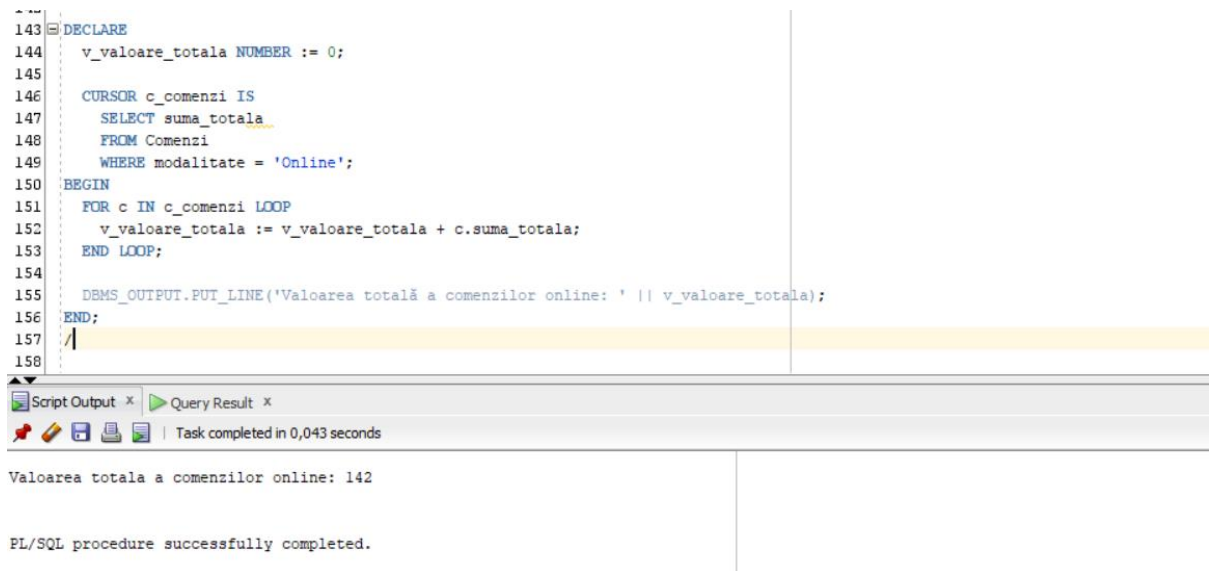
```
v_valoare_totala NUMBER := 0;
```

```
CURSOR c_comenzi IS
```

```

SELECT suma_totala
FROM Comenzi
WHERE modalitate = 'Online';
BEGIN
FOR c IN c_comenzi LOOP
    v_valoare_totala := v_valoare_totala + c.suma_totala;
END LOOP;
DBMS_OUTPUT.PUT_LINE('Valoarea totală a comenzilor online: ' || v_valoare_totala);
END;
/

```



The screenshot shows the Oracle SQL Developer interface. The main window displays a PL/SQL script with line numbers 143 to 158. The script declares a variable `v_valoare_totala` as a NUMBER, defines a cursor `c_comenzi` to select `suma_totala` from the `Comenzi` table where `modalitate` is 'Online', and then uses a loop to calculate the total value. The script ends with a `DBMS_OUTPUT.PUT_LINE` statement to display the result. Below the script, the 'Script Output' tab shows the execution results: 'Valoarea totala a comenzilor online: 142' and 'PL/SQL procedure successfully completed.' The 'Query Result' tab is also visible but empty.

```

143 DECLARE
144     v_valoare_totala NUMBER := 0;
145
146     CURSOR c_comenzi IS
147         SELECT suma_totala
148         FROM Comenzi
149         WHERE modalitate = 'Online';
150 BEGIN
151     FOR c IN c_comenzi LOOP
152         v_valoare_totala := v_valoare_totala + c.suma_totala;
153     END LOOP;
154
155     DBMS_OUTPUT.PUT_LINE('Valoarea totală a comenzilor online: ' || v_valoare_totala);
156 END;
157 /
158

```

Script Output x Query Result x

Task completed in 0,043 seconds

Valoarea totala a comenzilor online: 142

PL/SQL procedure successfully completed.

2. Să se afișeze lista cu prenumele, salariul și email-ul angajaților cu funcția de barista folosind un cursor explicit.

```

DECLARE

CURSOR c_angajati IS

    SELECT dp.prenume, a.salariul, dp.email
    FROM Angajati a
    JOIN Date_persoane dp ON a.CNP = dp.CNP
    WHERE a.functie = 'BARISTA';

v_prenume Date_persoane.prenume%TYPE;
v_salariul Angajati.salariul%TYPE;
v_email Date_persoane.email%TYPE;

```

```

BEGIN

OPEN c_angajati;

LOOP

    FETCH c_angajati INTO v_prenume, v_salariul, v_email;

    EXIT WHEN c_angajati%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE('Prenume: ' || v_prenume);

    DBMS_OUTPUT.PUT_LINE('Salariu: ' || v_salariul);

    DBMS_OUTPUT.PUT_LINE('Email: ' || v_email);

    DBMS_OUTPUT.PUT_LINE('-----');

END LOOP;

CLOSE c_angajati;

END;

/

```

```

163      JOIN Date_persoane dp ON a.CNP = dp.CNP
164      WHERE a.functie = 'BARISTA';
165
166      v_prenume Date_persoane.prenume%TYPE;
167      v_salariul Angajati.salariul%TYPE;
168      v_email Date_persoane.email%TYPE;
169  BEGIN
170      OPEN c_angajati;
171
172  LOOP
173      FETCH c_angajati INTO v_prenume, v_salariul, v_email;
174      EXIT WHEN c_angajati%NOTFOUND;
175
176      DBMS_OUTPUT.PUT_LINE('Prenume: ' || v_prenume);
177      DBMS_OUTPUT.PUT_LINE('Salariu: ' || v_salariul);
178      DBMS_OUTPUT.PUT_LINE('Email: ' || v_email);
179      DBMS_OUTPUT.PUT_LINE('-----');
180  END LOOP;
181
182  CLOSE c_angajati;
183  END;
184  /
185
186

```

Script Output x Query Result x

Task completed in 0,049 seconds

```

Prenume: Luna
Salariu: 2415
Email: luna@gmail.com
-----
Prenume: Pavel
Salariu: 2888
Email: ipavel@gmail.com

```

3. Să se afișeze lista cu prenumele, varsta și numărul de telefon clienților cu mai mult de 2 comenzi plasate folosind un cursor explicit.

DECLARE

CURSOR c\_clienti IS

SELECT d.prenume, d.varsta, d.nr\_telefon

```

FROM Date_persoane d
JOIN Clienti c ON d.CNP = c.CNP
WHERE c.nr_comenzi > 2;

```

```

v_prenume Date_persoane.prenume%TYPE;
v_varsta Date_persoane.varsta%TYPE;
v_nr_telefon Date_persoane.nr_telefon%TYPE;

```

```
BEGIN
```

```
OPEN c_clienti;
```

```
LOOP
```

```
FETCH c_clienti INTO v_prenume, v_varsta, v_nr_telefon;
```

```
EXIT WHEN c_clienti%NOTFOUND;
```

```
DBMS_OUTPUT.PUT_LINE('Prenume: ' || v_prenume);
```

```
DBMS_OUTPUT.PUT_LINE('Varsta: ' || v_varsta);
```

```
DBMS_OUTPUT.PUT_LINE('Numar telefon: ' || v_nr_telefon);
```

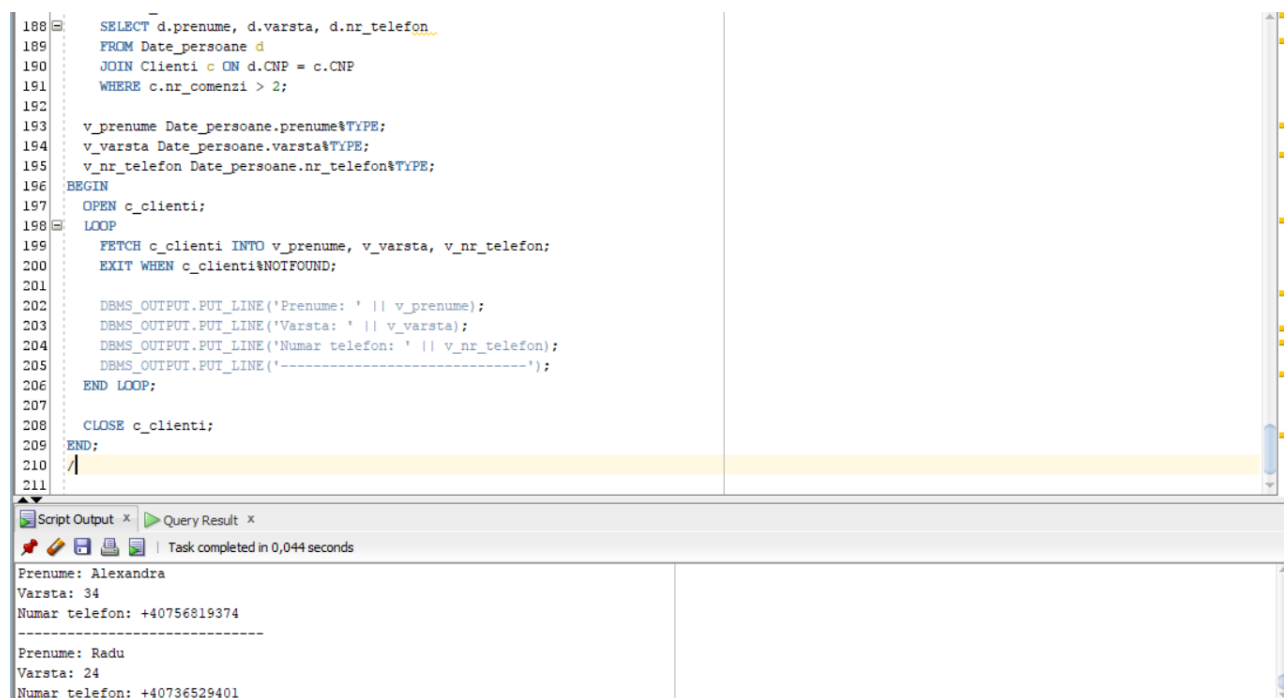
```
DBMS_OUTPUT.PUT_LINE('-----');
```

```
END LOOP;
```

```
CLOSE c_clienti;
```

```
END;
```

```
/
```



```

188 SELECT d.prenume, d.varsta, d.nr_telefon
189 FROM Date_persoane d
190 JOIN Clienti c ON d.CNP = c.CNP
191 WHERE c.nr_comenzi > 2;
192
193 v_prenume Date_persoane.prenume%TYPE;
194 v_varsta Date_persoane.varsta%TYPE;
195 v_nr_telefon Date_persoane.nr_telefon%TYPE;
196 BEGIN
197 OPEN c_clienti;
198 LOOP
199 FETCH c_clienti INTO v_prenume, v_varsta, v_nr_telefon;
200 EXIT WHEN c_clienti%NOTFOUND;
201
202 DBMS_OUTPUT.PUT_LINE('Prenume: ' || v_prenume);
203 DBMS_OUTPUT.PUT_LINE('Varsta: ' || v_varsta);
204 DBMS_OUTPUT.PUT_LINE('Numar telefon: ' || v_nr_telefon);
205 DBMS_OUTPUT.PUT_LINE('-----');
206 END LOOP;
207
208 CLOSE c_clienti;
209 END;
210 /
211

```

Script Output x Query Result x

Task completed in 0,044 seconds

Prenume: Alexandra Varsta: 34 Numar telefon: +40756819374 -----	
Prenume: Radu Varsta: 24 Numar telefon: +40736529401	

## Cursor implicit

1. Să se afișeze toate comenzile cu o sumă totală mai mare de 20 și să se calculeze suma totală a acestor comenzi.

DECLARE

```
v_suma_totala NUMBER := 0;
```

BEGIN

```
-- Deschideți cursorul implicit
```

```
FOR c IN (SELECT * FROM Comenzi WHERE suma_totala > 20) LOOP
```

```
    DBMS_OUTPUT.PUT_LINE('ID_comanda: ' || c.ID_comanda);
```

```
    DBMS_OUTPUT.PUT_LINE('Modalitate: ' || c.modalitate);
```

```
    DBMS_OUTPUT.PUT_LINE('Data comanda: ' || TO_CHAR(c.data_comanda, 'DD-MON-YYYY'));
```

```
    DBMS_OUTPUT.PUT_LINE('Stare comanda: ' || c.stare_comanda);
```

```
    DBMS_OUTPUT.PUT_LINE('Suma totala: ' || c.suma_totala);
```

```
    DBMS_OUTPUT.PUT_LINE('ID_produs: ' || c.ID_produs);
```

```
    DBMS_OUTPUT.PUT_LINE('-----');
```

```
    v_suma_totala := v_suma_totala + c.suma_totala;
```

```
END LOOP;
```

```
    DBMS_OUTPUT.PUT_LINE('Suma totala a comenzilor cu suma mai mare de 20: ' ||  
v_suma_totala);
```

```
END;
```

```
/
```



The screenshot displays the Oracle SQL Developer environment. The main window shows a PL/SQL script with line numbers 228 to 246. The script declares a variable `v_suma_totala` of type `NUMBER` and initializes it to 0. It then enters a `BEGIN` block where it uses an implicit cursor `FOR c IN (SELECT * FROM Comenzi WHERE suma_totala > 20) LOOP`. Inside the loop, it prints several details for each command: `ID_comanda`, `Modalitate`, `Data comanda` (formatted as DD-MON-YYYY), `Stare comanda`, `Suma totala`, and `ID_produs`, separated by a dashed line. It also accumulates the total sum in `v_suma_totala`. After the loop, it prints the final sum. The script ends with `END;` and a slash `/` to execute. Below the script editor, the 'Script Output' window shows the results of the execution, including the command details for ID 8 and the final sum of 131.

```
228 DECLARE
229     v_suma_totala NUMBER := 0;
230 BEGIN
231     -- Deschideți cursorul implicit
232     FOR c IN (SELECT * FROM Comenzi WHERE suma_totala > 20) LOOP
233         DBMS_OUTPUT.PUT_LINE('ID_comanda: ' || c.ID_comanda);
234         DBMS_OUTPUT.PUT_LINE('Modalitate: ' || c.modalitate);
235         DBMS_OUTPUT.PUT_LINE('Data comanda: ' || TO_CHAR(c.data_comanda, 'DD-MON-YYYY'));
236         DBMS_OUTPUT.PUT_LINE('Stare comanda: ' || c.stare_comanda);
237         DBMS_OUTPUT.PUT_LINE('Suma totala: ' || c.suma_totala);
238         DBMS_OUTPUT.PUT_LINE('ID_produs: ' || c.ID_produs);
239         DBMS_OUTPUT.PUT_LINE('-----');
240
241         v_suma_totala := v_suma_totala + c.suma_totala;
242     END LOOP;
243     DBMS_OUTPUT.PUT_LINE('Suma totala a comenzilor cu suma mai mare de 20: ' || v_suma_totala);
244 END;
245 /
246
```

Script Output x Query Result x

Task completed in 0,114 seconds

```
ID_comanda: 8
Modalitate: Online
Data comanda: 17-IAN-2023
Stare comanda: livrata
Suma totala: 39
ID_produs: 2
-----
Suma totala a comenzilor cu suma mai mare de 20: 131
```

2. Să se afișeze toate locațiile dintr-un anumit oraș și să se numere câte locații există în acel oraș.

DECLARE

v\_oras\_cautat VARCHAR2(20) := 'Bucuresti';

v\_numar\_locatii NUMBER := 0;

BEGIN

-- Deschideți cursorul implicit

FOR c IN (SELECT \* FROM Locatii WHERE oras = v\_oras\_cautat) LOOP

DBMS\_OUTPUT.PUT\_LINE('ID\_locatie: ' || c.ID\_locatie);

DBMS\_OUTPUT.PUT\_LINE('Oras: ' || c.oras);

DBMS\_OUTPUT.PUT\_LINE('Zona: ' || c.zona);

DBMS\_OUTPUT.PUT\_LINE('-----');

v\_numar\_locatii := v\_numar\_locatii + 1;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Numarul de locatii in orasul ' || v\_oras\_cautat || ': ' ||  
v\_numar\_locatii);

END;

/



The screenshot shows the Oracle SQL Developer interface. The main window displays a PL/SQL script with line numbers 249 to 265. The script declares a variable v\_oras\_cautat with the value 'Bucuresti' and a counter v\_numar\_locatii. It then uses a FOR loop to iterate over the 'Locatii' table, printing the ID, city, and zone of each location. After the loop, it prints the total count of locations in the specified city. The script is executed, and the 'Script Output' window at the bottom shows the results: three locations in Bucuresti (ID 5, 6, 7) and a total count of 3.

```
249 DECLARE
250     v_oras_cautat VARCHAR2(20) := 'Bucuresti';
251     v_numar_locatii NUMBER := 0;
252 BEGIN
253     -- Deschideți cursorul implicit
254     FOR c IN (SELECT * FROM Locatii WHERE oras = v_oras_cautat) LOOP
255         DBMS_OUTPUT.PUT_LINE('ID_locatie: ' || c.ID_locatie);
256         DBMS_OUTPUT.PUT_LINE('Oras: ' || c.oras);
257         DBMS_OUTPUT.PUT_LINE('Zona: ' || c.zona);
258         DBMS_OUTPUT.PUT_LINE('-----');
259         v_numar_locatii := v_numar_locatii + 1;
260     END LOOP;
261
262     DBMS_OUTPUT.PUT_LINE('Numarul de locatii in orasul ' || v_oras_cautat || ': ' || v_numar_locatii);
263 END;
264 /
265
```

Script Output x Query Result x

Task completed in 0,053 seconds

-----  
ID\_locatie: 5  
Oras: Bucuresti  
Zona: Victoriei  
-----  
Numarul de locatii in orasul Bucuresti: 3

3. Să se afișeze toate produsele cu prețul mai mare de 15 și să se calculeze prețul total al acestora.

DECLARE

v\_pret\_total NUMBER := 0;

BEGIN

```

FOR c IN (SELECT * FROM Produse WHERE pret > 15) LOOP
    -- Afișați informațiile despre produs
    DBMS_OUTPUT.PUT_LINE('ID_produs: ' || c.ID_produs);
    DBMS_OUTPUT.PUT_LINE('Denumire: ' || c.denumire);
    DBMS_OUTPUT.PUT_LINE('Pret: ' || c.pret);
    DBMS_OUTPUT.PUT_LINE('-----');

    v_pret_total := v_pret_total + c.pret;
END LOOP;

DBMS_OUTPUT.PUT_LINE('Pretul total al produselor cu pretul mai mare de 15: ' || v_pret_total);
END;
/

```

```

268 DECLARE
269     v_pret_total NUMBER := 0;
270 BEGIN
271     FOR c IN (SELECT * FROM Produse WHERE pret > 15) LOOP
272         -- Afișați informațiile despre produs
273         DBMS_OUTPUT.PUT_LINE('ID_produs: ' || c.ID_produs);
274         DBMS_OUTPUT.PUT_LINE('Denumire: ' || c.denumire);
275         DBMS_OUTPUT.PUT_LINE('Pret: ' || c.pret);
276         DBMS_OUTPUT.PUT_LINE('-----');
277         v_pret_total := v_pret_total + c.pret;
278     END LOOP;
279     DBMS_OUTPUT.PUT_LINE('Pretul total al produselor cu pretul mai mare de 15: ' || v_pret_total);
280 END;
281 /

```

Script Output x Query Result x

Task completed in 0,038 seconds

```

Pret: 20
-----
Pretul total al produselor cu pretul mai mare de 15: 73

PL/SQL procedure successfully completed.

```

## F. Funcții, proceduri și pachete

### Funcții

1. Creați o funcție care să returneze numărul total de angajați care au salariul mai mare decât salariul mediu.

```
CREATE OR REPLACE FUNCTION NumarAngajatiCuSalariuMaiMare RETURN NUMBER IS
```

```
    v_salariu_mediu NUMBER(6);
```

```
    v_numar_angajati NUMBER(10);
```

```
BEGIN
```

```
    SELECT AVG(salariul)
```



```
INTO v_salariu_mediu
```

```
FROM Angajati;
```

```
SELECT COUNT(*)
```

```
INTO v_numar_angajati
```

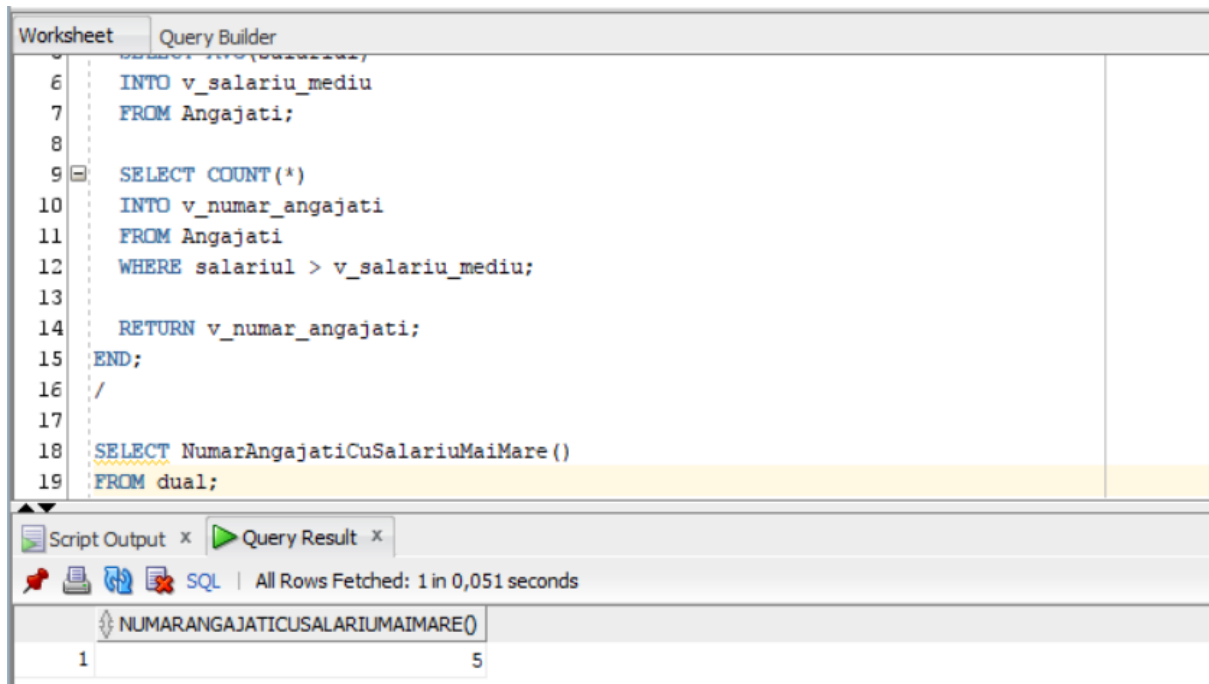
```
FROM Angajati
```

```
WHERE salariul > v_salariu_mediu;
```

```
RETURN v_numar_angajati;
```

```
END;
```

```
/
```



2. Calculează suma totală a comenzilor într-o anumită stare.

```
CREATE OR REPLACE FUNCTION CalculSumaTotala(stare_in VARCHAR2) RETURN  
NUMBER IS
```

```
v_suma_totala NUMBER(10) := 0;
```

```
BEGIN
```

```
SELECT SUM(suma_totala)
```

```
INTO v_suma_totala
```

```
FROM Comenzi
```

```
WHERE stare_comanda = stare_in;
```

```

RETURN v_suma_totala;

END;

/

DECLARE

v_suma NUMBER(10);

BEGIN

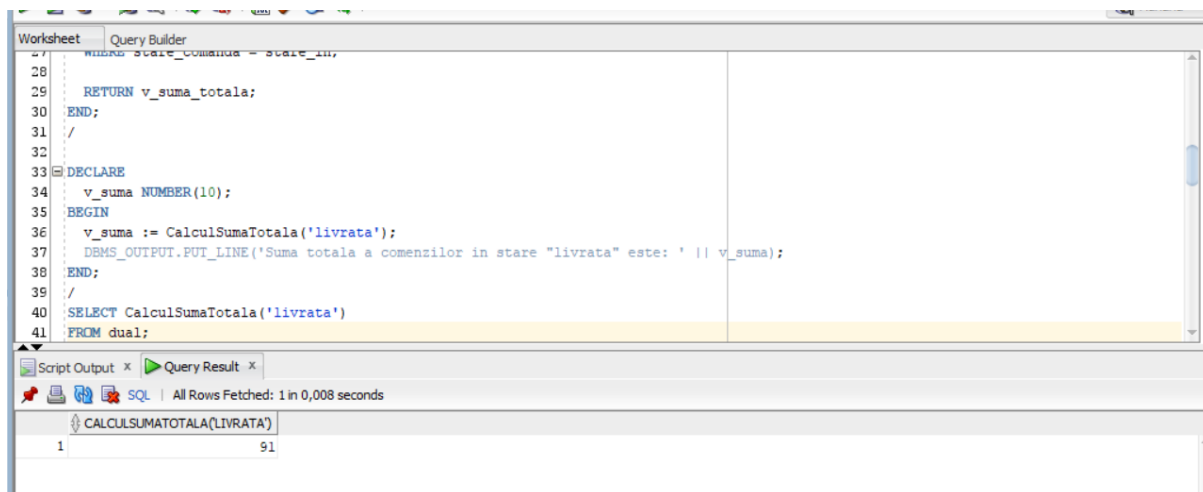
v_suma := CalculSumaTotala('livrata');

DBMS_OUTPUT.PUT_LINE('Suma totala a comenzilor in stare "livrata" este: ' || v_suma);

END;

/

```



3. Returnează numărul total de produse cu prețul mai mare decât o valoare specificată.

```

CREATE OR REPLACE FUNCTION NumarProduse(valoare_pret NUMBER) RETURN NUMBER
IS

```

```

v_numar_produse NUMBER(10) := 0;

BEGIN

SELECT COUNT(*)

INTO v_numar_produse

FROM Produse

WHERE pret > valoare_pret;

RETURN v_numar_produse;

END;

/

```

DECLARE

v\_numar NUMBER(10);

BEGIN

v\_numar := NumarProduse(15);

DBMS\_OUTPUT.PUT\_LINE('Numarul de produse cu pretul mai mare de 15 este: ' || v\_numar);

END;

/



## Proceduri

1. Creati o procedura care primeste ca parametru de intrare id-ul unui client si returneaza prin parametru de tip out email-ul acestuia.

CREATE OR REPLACE PROCEDURE EmailClient(

p\_ID\_client IN VARCHAR2,

p\_email OUT VARCHAR2

) IS

BEGIN

SELECT d.email

INTO p\_email

FROM Angajati a

JOIN Date\_persoane d ON a.CNP = d.CNP

WHERE a.ID\_angajat = p\_ID\_client;

END;

/

DECLARE

v\_ID\_client VARCHAR2(2) := &id;

v\_email VARCHAR2(30);

BEGIN

EmailClient(v\_ID\_client, v\_email);

DBMS\_OUTPUT.PUT\_LINE('Email-ul clientului cu ID-ul ' || v\_ID\_client || ' este: ' || v\_email);

END;

/

2. Actualizează starea comenzilor în funcție de suma totală.

CREATE OR REPLACE PROCEDURE ActualizareStareComenzi AS

BEGIN

UPDATE Comenzi

SET stare\_comanda = CASE

WHEN suma\_totala > 20 THEN 'Livrata'

ELSE 'Pregatita'

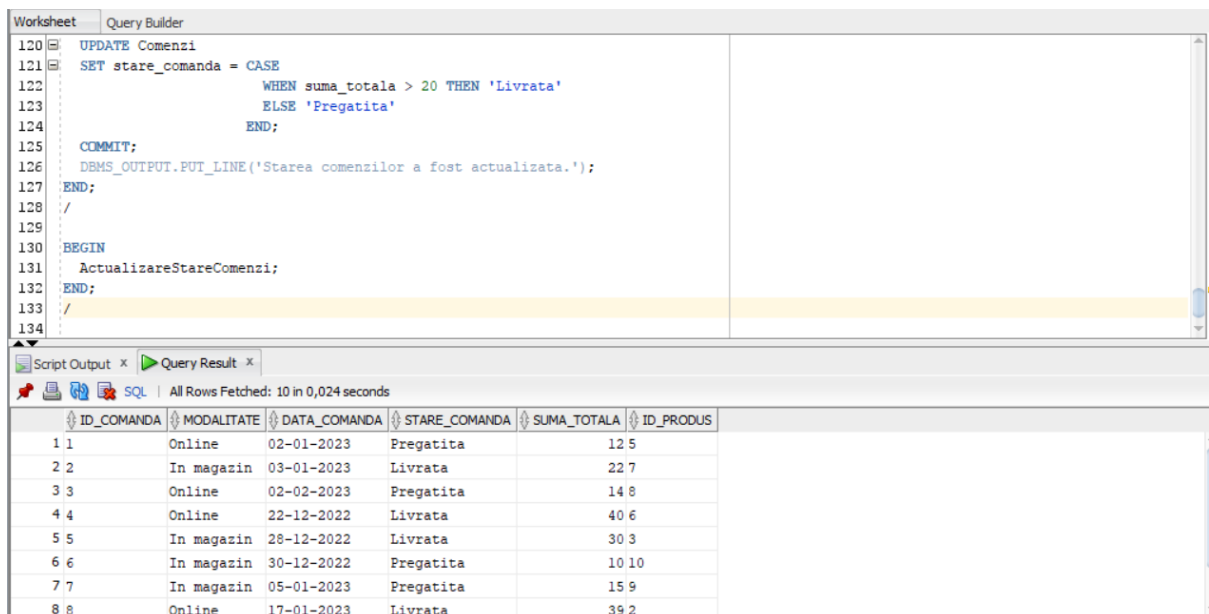
END;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Starea comenzilor a fost actualizata.');

END;

/



The screenshot shows a SQL query editor with a 'Worksheet' tab. The query is a PL/SQL procedure named 'ActualizareStareComenzi' that updates the 'stare\_comanda' column of the 'Comenzi' table based on the 'suma\_totala'. The procedure uses a CASE statement to set the status to 'Livrata' if the total sum is greater than 20, and 'Pregatita' otherwise. It also includes a COMMIT statement and a DBMS\_OUTPUT.PUT\_LINE statement to confirm the update. The procedure is executed, and the results are shown in the 'Query Result' tab. The results table has 8 rows, each representing a command with its ID, modalitate, data, stare\_comanda, suma\_totala, and ID\_produs.

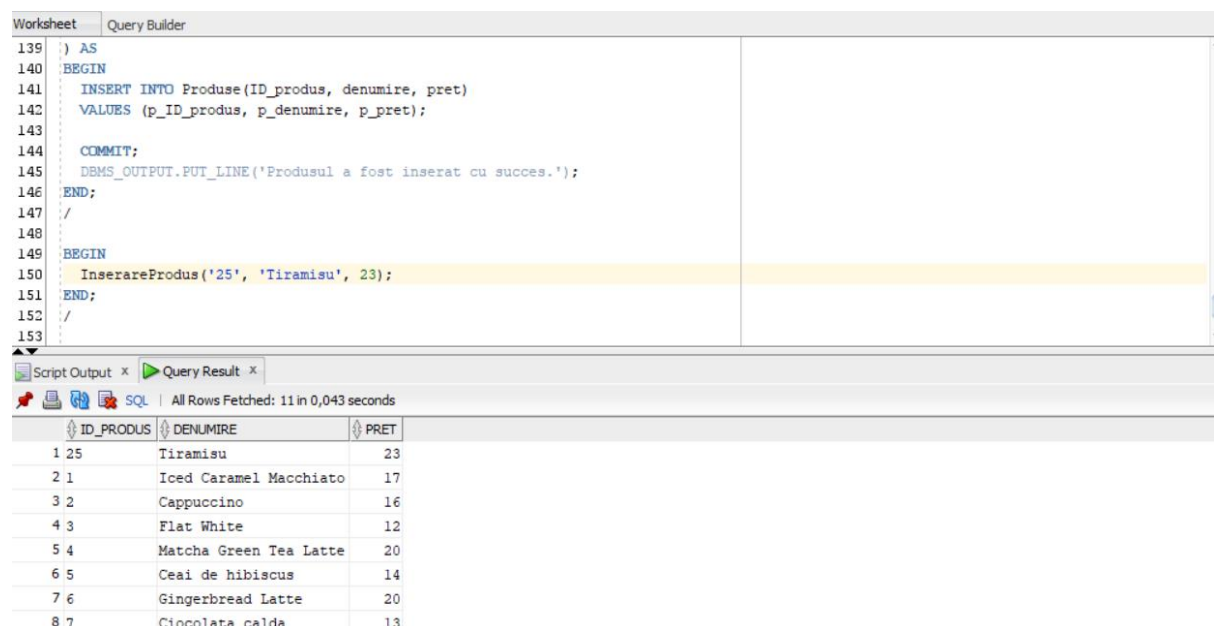
ID_COMANDA	MODALITATE	DATA_COMANDA	STARE_COMANDA	SUMA_TOTALA	ID_PRODUS
1 1	Online	02-01-2023	Pregatita	12 5	
2 2	In magazin	03-01-2023	Livrata	22 7	
3 3	Online	02-02-2023	Pregatita	14 8	
4 4	Online	22-12-2022	Livrata	40 6	
5 5	In magazin	28-12-2022	Livrata	30 3	
6 6	In magazin	30-12-2022	Pregatita	10 10	
7 7	In magazin	05-01-2023	Pregatita	15 9	
8 8	Online	17-01-2023	Livrata	39 2	

3. Sa se insereze un nou produs in tabela Produse.

```
CREATE OR REPLACE PROCEDURE InserareProdus(  
    p_ID_produs IN VARCHAR2,  
    p_denumire IN VARCHAR2,  
    p_pret IN NUMBER  
) AS  
BEGIN  
    INSERT INTO Produse(ID_produs, denumire, pret)  
    VALUES (p_ID_produs, p_denumire, p_pret);  
  
    COMMIT;  
  
    DBMS_OUTPUT.PUT_LINE('Produsul a fost inserat cu succes.');
```

END;

/



The screenshot shows a SQL IDE interface. The top pane displays a PL/SQL script with line numbers 139 to 153. The script defines a procedure `InserareProdus` and calls it with parameters '25', 'Tiramisu', and 23. The bottom pane shows the 'Query Result' tab with a table of 8 rows. The first row corresponds to the newly inserted product.

ID_PRODUS	DENUMIRE	PRET
1 25	Tiramisu	23
2 1	Iced Caramel Macchiato	17
3 2	Cappuccino	16
4 3	Flat White	12
5 4	Matcha Green Tea Latte	20
6 5	Ceai de hibiscus	14
7 6	Gingerbread Latte	20
8 7	Ciocolata calda	13

## Pachete

1. Pachetul conține două proceduri: una pentru a actualiza starea comenzilor și alta pentru a obține suma totală a comenzilor pentru un anumit produs.

```
CREATE OR REPLACE PACKAGE Pachet1 IS
```

```
    PROCEDURE ActualizareStareComenzi;
```

```
    PROCEDURE ObtinereSuma(ID_produs IN VARCHAR2, p_suma_totala OUT NUMBER);
```

END Pachet1;

/

CREATE OR REPLACE PACKAGE BODY Pachet1 IS

PROCEDURE ActualizareStareComenzi AS

BEGIN

UPDATE Comenzi

SET stare\_comanda = CASE

WHEN suma\_totala > 20 THEN 'Livrata'

ELSE 'Pregatita'

END;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Starea comenzilor a fost actualizata.');

END ActualizareStareComenzi;

PROCEDURE ObtinereSuma(ID\_produc IN VARCHAR2, p\_suma\_totala OUT NUMBER) AS

BEGIN

SELECT SUM(suma\_totala)

INTO p\_suma\_totala

FROM Comenzi

WHERE ID\_produc = ID\_produc;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

p\_suma\_totala := 0;

END ObtinereSuma;

END Pachet1;

/

BEGIN

Pachet1.ActualizareStareComenzi;

END;

/

DECLARE

v\_suma\_totala NUMBER;

BEGIN

Pachet1.ObtinereSuma('Tiramisu', v\_suma\_totala);

DBMS\_OUTPUT.PUT\_LINE('Suma totala pentru produsul P1 este: ' || v\_suma\_totala);

END;

/

2. Pachetul contine o functie care insereaza o noua locatie in tabela locatii si o procedura care actualizeaza locatia.

CREATE OR REPLACE PACKAGE Pachet2 IS

FUNCTION InserareLocatie(

p\_ID\_locatie IN Locatii.ID\_locatie%TYPE,

p\_oras IN Locatii.oras%TYPE,

p\_zona IN Locatii.zona%TYPE

) RETURN BOOLEAN;

PROCEDURE ActualizareLocatie(

p\_ID\_locatie IN Locatii.ID\_locatie%TYPE,

p\_oras IN Locatii.oras%TYPE,

p\_zona IN Locatii.zona%TYPE

);

END Pachet2;

/

CREATE OR REPLACE PACKAGE BODY Pachet2 IS

FUNCTION InserareLocatie(

p\_ID\_locatie IN Locatii.ID\_locatie%TYPE,

```

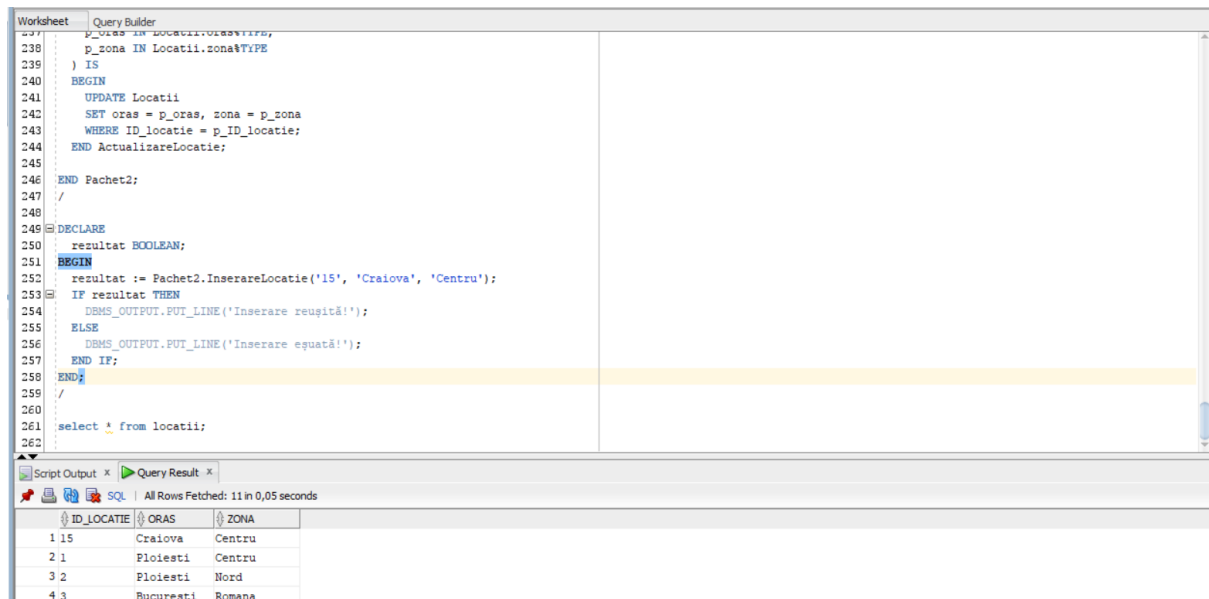
p_oras IN Locatii.oras%TYPE,
p_zona IN Locatii.zona%TYPE
) RETURN BOOLEAN IS
BEGIN
    INSERT INTO Locatii (ID_locatie, oras, zona)
    VALUES (p_ID_locatie, p_oras, p_zona);
    RETURN TRUE;
END InserareLocatie;

PROCEDURE ActualizareLocatie(
    p_ID_locatie IN Locatii.ID_locatie%TYPE,
    p_oras IN Locatii.oras%TYPE,
    p_zona IN Locatii.zona%TYPE
) IS
BEGIN
    UPDATE Locatii
    SET oras = p_oras, zona = p_zona
    WHERE ID_locatie = p_ID_locatie;
END ActualizareLocatie;

END Pachet2;

/

```



The screenshot shows a SQL development environment with a script editor and a query result table.

**Script Editor:** The script contains the following SQL code:

```

237 p_oras IN Locatii.oras%TYPE,
238 p_zona IN Locatii.zona%TYPE
239 ) IS
240 BEGIN
241     UPDATE Locatii
242     SET oras = p_oras, zona = p_zona
243     WHERE ID_locatie = p_ID_locatie;
244 END ActualizareLocatie;
245
246 END Pachet2;
247 /
248
249 DECLARE
250     rezultat BOOLEAN;
251 BEGIN
252     rezultat := Pachet2.InserareLocatie('15', 'Craiova', 'Centru');
253 IF rezultat THEN
254     DBMS_OUTPUT.PUT_LINE('Inserare reusita!');
255 ELSE
256     DBMS_OUTPUT.PUT_LINE('Inserare eșuat!');
257 END IF;
258 END;
259 /
260 select * from locatii;
261
262

```

**Query Result:** The query result table shows the following data:

ID_LOCATIE	ORAS	ZONA
1 15	Craiova	Centru
2 1	Ploiesti	Centru
3 2	Ploiesti	Nord
4 3	Bucuresti	Romana



```

261 BEGIN
262   Pachet2.ActualizareLocatie('15', 'Timisoara', 'Nord');
263   DBMS_OUTPUT.PUT_LINE('Actualizare reușită. ');
264 END;
265 /
266
267 select * from locatii;
268

```

Script Output x Query Result x

SQL | All Rows Fetched: 11 in 0,005 seconds

	ID_LOCATIE	ORAS	ZONA
1	15	Timisoara	Nord
2	1	Ploiesti	Centru
3	2	Ploiesti	Nord
4	3	Bucuresti	Romana

## G. Declanșatori

1. Sa se creeze un trigger care sa intoarca o eroare atunci cand se incearca inserarea unui produs pentru care pretul este mai mare de 50.

CREATE OR REPLACE TRIGGER verifica\_pret

before insert on PRODUSE

for each row

BEGIN

if :new.PRET > 50 then

RAISE\_APPLICATION\_ERROR(-20001, 'Pretul trebuie sa fie mai mic sau egal cu 50.');

end if;

END;

/

```

1 CREATE OR REPLACE TRIGGER verifica_pret
2 before insert on PRODUSE
3 for each row
4 BEGIN
5   if :new.PRET > 50 then
6     RAISE_APPLICATION_ERROR(-20001, 'Pretul trebuie sa fie mai mic sau egal cu 50. ');
7   end if;
8 END;
9 /
10
11 insert into produse(id_produ, denumire, pret)
12 values (13, 'Fursec', 55);
13

```

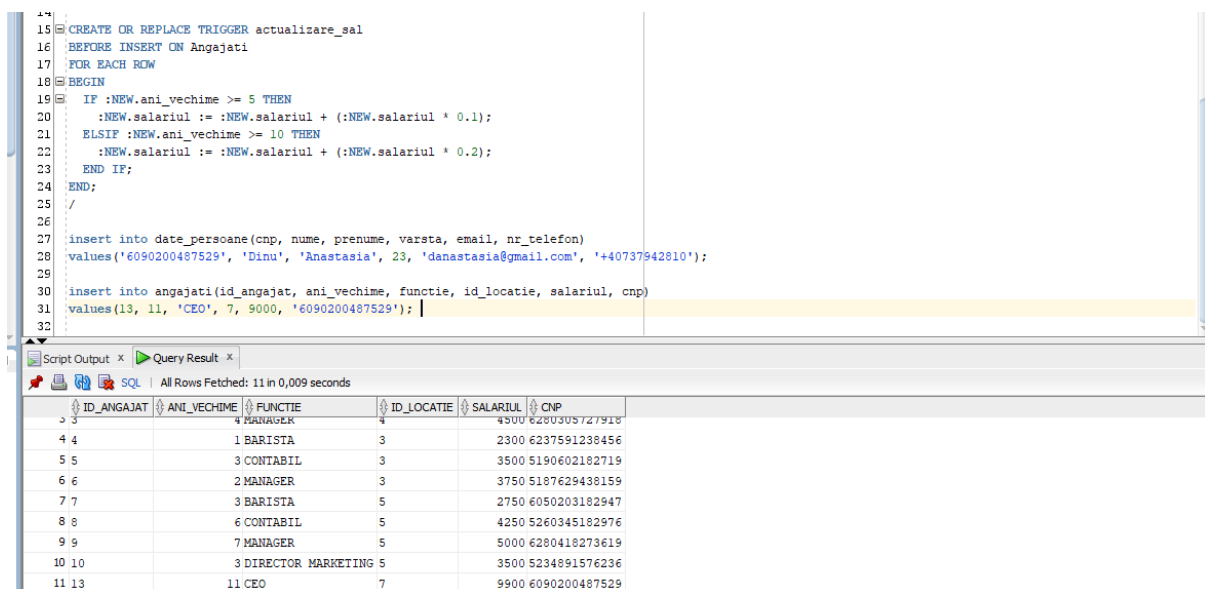
Script Output x

Task completed in 0,029 seconds

Error report -  
ORA-20001: Pretul trebuie sa fie mai mic sau egal cu 50.  
ORA-06512: at "MOCANUV\_56.VERIFICA\_PRET", line 3  
ORA-04088: error during execution of trigger 'MOCANUV\_56.VERIFICA\_PRET'

2. Sa se creeze un trigger care sa mareasca salariul in functie de anii de vechime la fiecare insert in tabela Angajati.

```
CREATE OR REPLACE TRIGGER actualizare_sal
BEFORE INSERT ON Angajati
FOR EACH ROW
BEGIN
    IF :NEW.ani_vechime >= 5 THEN
        :NEW.salariul := :NEW.salariul + (:NEW.salariul * 0.1);
    ELSIF :NEW.ani_vechime >= 10 THEN
        :NEW.salariul := :NEW.salariul + (:NEW.salariul * 0.2);
    END IF;
END;
/
```



The screenshot displays a SQL script in an IDE. The script creates a trigger named 'actualizare\_sal' that increases the salary of new hires based on their years of experience. It then inserts a new employee into the 'data\_persoane' table and a new entry into the 'angajati' table. Below the script, the 'Query Result' pane shows the data in the 'angajati' table after the insert operation.

ID_ANGAJAT	ANI_VECHIME	FUNCTIE	ID_LOCATIE	SALARIUL	CNP
3	3	4 MANAGER	4	4500	6200305727918
4	4	1 BARISTA	3	2300	6237591238456
5	5	3 CONTABIL	3	3500	5190602182719
6	6	2 MANAGER	3	3750	5187629438159
7	7	3 BARISTA	5	2750	6050203182947
8	8	6 CONTABIL	5	4250	5260345182976
9	9	7 MANAGER	5	5000	6280418273619
10	10	3 DIRECTOR MARKETING	5	3500	5234891576236
11	13	11 CEO	7	9900	6090200487529

3. Sa se creeze un trigger care sa se activeze la actualizarea starii unei comenzi si sa actualizeze automat data\_comanda cu data curenta.

```
CREATE OR REPLACE TRIGGER actualizare_data
BEFORE UPDATE ON Comenzi
FOR EACH ROW
BEGIN
    IF :NEW.stare_comanda <> :OLD.stare_comanda THEN
        :NEW.data_comanda := SYSDATE;
    END IF;
END;
/
```

/

Worksheet

Query Builder

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

```
CREATE OR REPLACE TRIGGER actualizar_data
BEFORE UPDATE ON Comenzi
FOR EACH ROW
BEGIN
    IF :NEW.stare_comanda <> :OLD.stare_comanda THEN
        :NEW.data_comanda := SYSDATE;
    END IF;
END;
/

INSERT INTO Comenzi (ID_comanda, modalitate, data_comanda, stare_comanda, suma_totala, ID_produs)
VALUES ('20', 'Online', TO_DATE('01-MAY-2023', 'DD-MON-YYYY'), 'Livrata', 54, '4');

SELECT * FROM Comenzi;

UPDATE Comenzi SET stare_comanda = 'Pregatita' WHERE ID_comanda = '20';
```

Script Output x

Query Result x

SQL

All Rows Fetched: 11 in 0,005 seconds

ID_COMANDA	MODALITATE	DATA_COMANDA	STARE_COMANDA	SUMA_TOTALA	ID_PRODUS
2 2	In magazin	03-JAN-23	Livrata	22 7	
3 3	Online	02-FEB-23	Pregatita	14 8	
4 4	Online	22-DEC-22	Livrata	40 6	
5 5	In magazin	28-DEC-22	Livrata	30 3	
6 6	In magazin	30-DEC-22	Pregatita	10 10	
7 7	In magazin	05-JAN-23	Pregatita	15 9	
8 8	Online	17-JAN-23	Livrata	39 2	
9 9	Online	28-JAN-23	Pregatita	17 1	
10 10	Online	03-FEB-23	Pregatita	20 4	
11 20	Online	23-MAY-23	Pregatita	54 4	