

Proiect
Sisteme de Gestiune a Bazelor de Date

GESTIUNEA UNUI RESTAURANT

Guțu Ștefania-Alexandra

grupa 241

ianuarie 2022

Cuprins

1	Prezentarea bazei de date	1
2	Diagrama entitate-relație (ERD)	2
3	Diagrama conceptuală	3
4	Implementarea diagramei conceptuale în Oracle	4
5	Adăugarea informațiilor în tabele	8
6	Subprogram stocat care utilizează două tipuri de colecție	21
7	Subprogram stocat care utilizează un cursor	27
8	Subprogram stocat de tip funcție care utilizează 3 tabele într-o comandă SQL	32
9	Subprogram stocat de tip procedură care utilizează 5 tabele într-o comandă SQL	38
10	Trigger de tip LMD la nivel de comandă	43
11	Trigger de tip LMD la nivel de linie	46
12	Trigger de tip LDD	52
13	Pachet care conține obiectele definite anterior	54
14	Pachet care include tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate	66

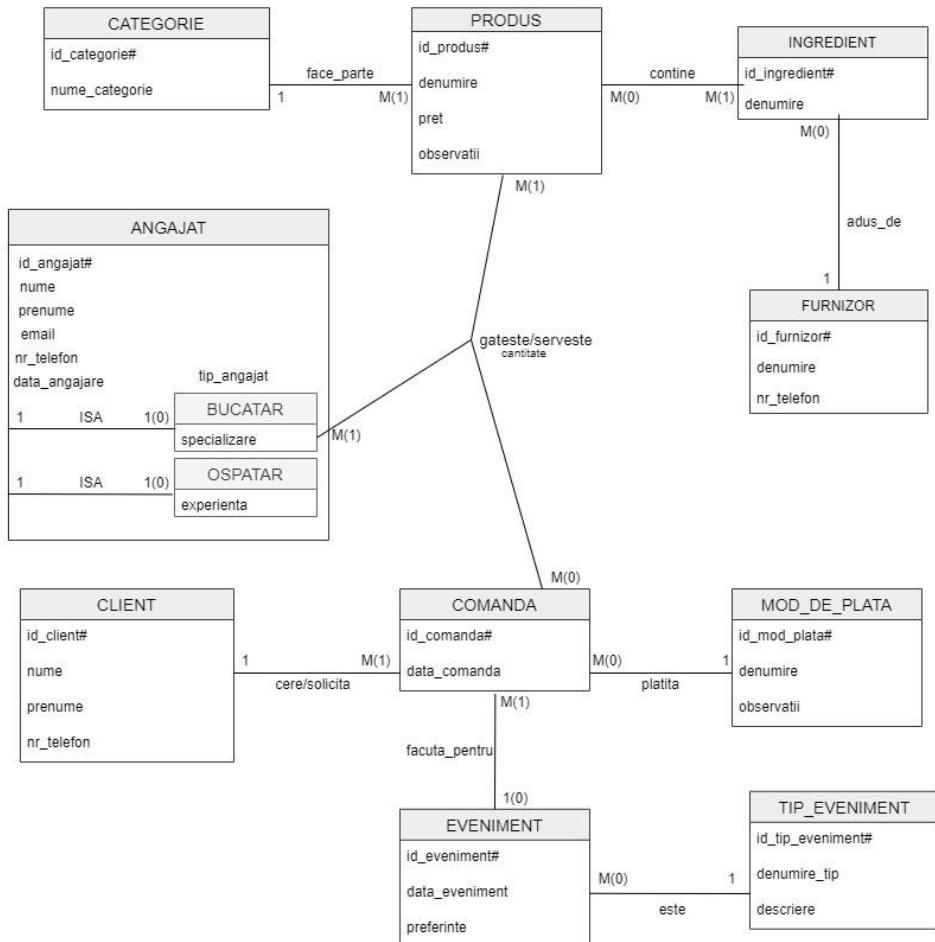
1. Prezentarea bazei de date

Proiectul cuprinde proiectarea unei baze de date ce furnizează informații despre un restaurant, permitând o bună gestionare a comenziilor, clientilor, evenimentelor, angajaților, produselor și a furnizorilor.

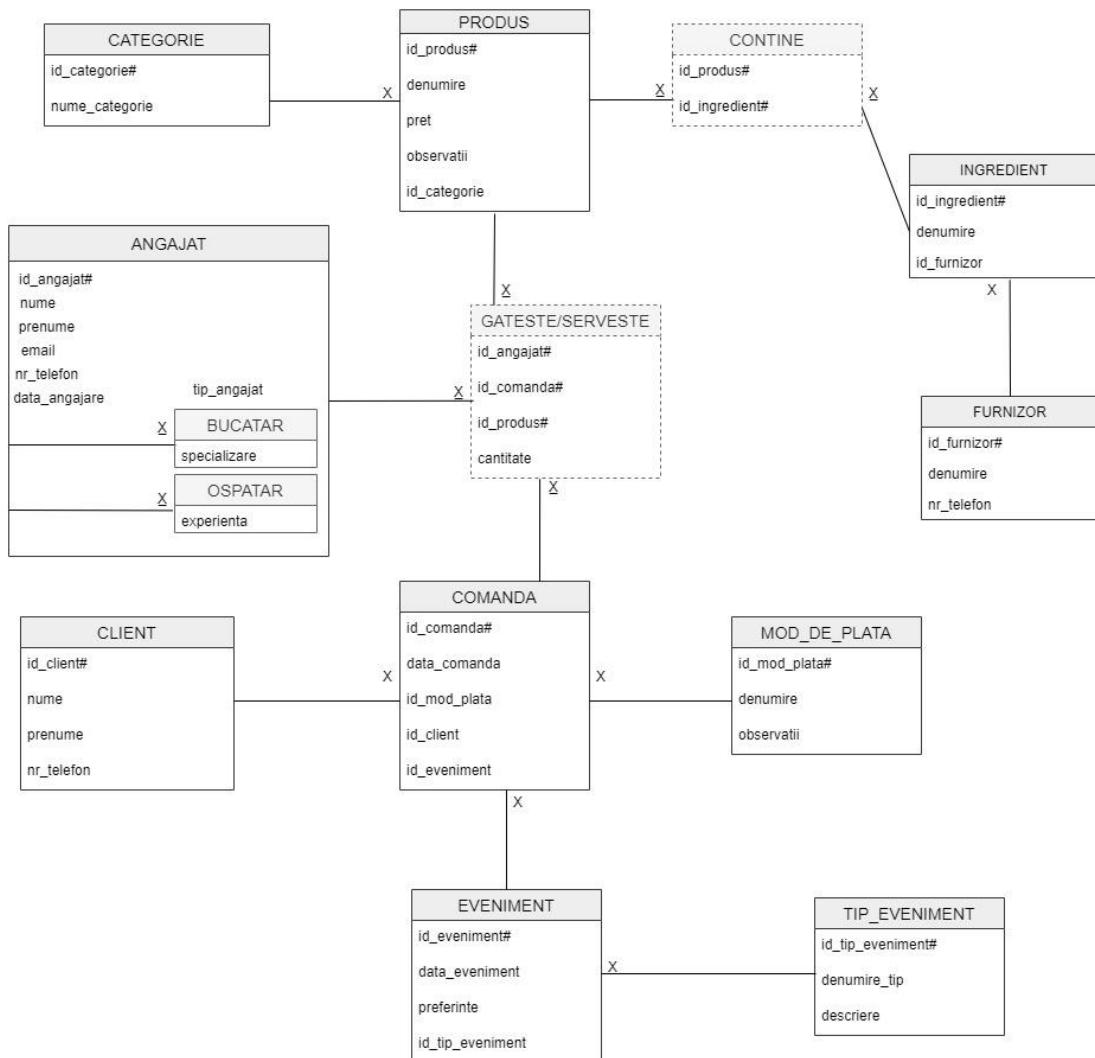
Tema aleasă are ca scop gestiunea tuturor datelor utile ce au în centru comenziile date, pentru o administrare și o organizare eficientă a restaurantului.

Astfel, comenzi date pot fi ușor gestionate, deoarece fiecare este atribuită unui client, în unele cazuri și pentru un eveniment. De asemenea, sunt stocate și informații despre produsele comandate, categoria și ingredientele acestora, cât și despre furnizorii ingredientelor. Putem gestiona cu ușurință și angajații restaurantului (bucătari și ospătari) alături de produsele gătite/servite de fiecare.

2. Diagrama entitate-relație (ERD)



3. Diagrama conceptuală



4. Implementarea diagramei conceptuale în Oracle

Crearea tabelului *clienti*:

```

1 CREATE TABLE clienti
2   (id_client number(5),
3    nume varchar2(25) constraint c_nume not null,
4    prenume varchar2(25),
5    nr_telefon char(10),
6    CONSTRAINT client_pk PRIMARY KEY (id_client),
7    unique(nr_telefon)
8 );

```

Crearea tabelului *tipuri_evenimente*:

```

1 CREATE TABLE tipuri_evenimente
2   (id_tip_eveniment number(5),
3    denumire_tip varchar2(20),
4    descriere long,
5    CONSTRAINT tip_eveniment_pk PRIMARY KEY (id_tip_eveniment)
6 );

```

Crearea tabelului *evenimente*:

```

1 CREATE TABLE evenimente
2   (id_eveniment number(5),
3    data_eveniment date,
4    preferinte long,
5    id_tip_eveniment number(5) CONSTRAINT c_tip_eveniment not null,
6    CONSTRAINT eveniment_pk PRIMARY KEY (id_eveniment),
7    CONSTRAINT eveniment_fk foreign key(id_tip_eveniment) references
8      tipuri_evenimente(id_tip_eveniment)
9 );

```

Crearea tabelului *moduri_de_plata*:

```

1 CREATE TABLE moduri_de_plata
2   (id_mod_plata number(5),
3    denumire varchar2(20),
4    observatii long,
5    CONSTRAINT mod_plata_pk PRIMARY KEY (id_mod_plata),

```

```

6    unique(denumire)
7 );

```

Crearea tabelului *comenzi*:

```

1 CREATE TABLE comenzi
2   (id_comanda number(5),
3   data_comanda date default sysdate,
4   id_mod_plata number(5) constraint mod_not_null not null,
5   id_client number(5) constraint client_not_null not null,
6   id_eveniment number(5),
7   CONSTRAINT comenzi_pk PRIMARY KEY (id_comanda),
8   CONSTRAINT comanda_mod_plata_fk foreign key(id_mod_plata) references
      moduri_de_plata(id_mod_plata),
9   CONSTRAINT comanda_client_fk foreign key(id_client) references
      clienti(id_client),
10  CONSTRAINT comanda_eveniment_fk foreign key(id_eveniment) references
      evenimente(id_eveniment)
11 );

```

Crearea tabelului *categorii*:

```

1 CREATE TABLE categorii
2   (id_categorie number(5),
3   nume_categorie varchar(20) constraint nume_categ_not_null not null,
4   CONSTRAINT categorii_pk PRIMARY KEY (id_categorie),
5   unique(nume_categorie)
6 );

```

Crearea tabelului *furnizori*:

```

1 CREATE TABLE furnizori
2   (id_furnizor number(5),
3   denumire varchar2(20) constraint denumiref_not_null not null,
4   nr_telefon char(10) constraint nr_telf_not_null not null,
5   unique(nr_telefon),
6   CONSTRAINT furnizori_pk PRIMARY KEY (id_furnizor)
7 );

```

Crearea tabelului *ingrediente*:

```

1 CREATE TABLE ingrediente
2   (id_ingredient number(5),
3   denumire varchar2(20) constraint denumire_i_not_null not null,
4   id_furnizor number(5) constraint furnizor_not_null not null,
5   CONSTRAINT ingrediente_pk PRIMARY KEY (id_ingredient),
6   CONSTRAINT ingred_furnizor_fk foreign key(id_furnizor) references
      furnizori(id_furnizor),

```

```

7||     unique(denumire)
8|| );

```

Crearea tabelului *produse*:

```

1| CREATE TABLE produse
2|   (id_produs number(5),
3|    denumire varchar2(30) constraint denumire_p_not_null not null,
4|    pret number(3) constraint pret_not_null not null,
5|    observatii long,
6|    id_categorie number(5) constraint categ_not_null not null,
7|    unique(denumire),
8|    check(pret>0),
9|    CONSTRAINT produse_pk PRIMARY KEY (id_produs),
10|   CONSTRAINT produs_categ_fk foreign key(id_categorie) references
11|     categorii(id_categorie)
11| );

```

Crearea tabelului *contine*:

```

1| CREATE TABLE contine
2|   (id_produs number(5) constraint produs_not_null not null,
3|    id_ingredient number(5) constraint ingred_not_null not null,
4|    CONSTRAINT contine_ingred_fk foreign key(id_ingredient) references
4|      ingrediente(id_ingredient),
5|    CONSTRAINT contine_produs_fk foreign key(id_produs) references
5|      produse(id_produs),
6|    CONSTRAINT pk_contine primary key(id_produs, id_ingredient)
7| );

```

Crearea tabelului *angajati*:

```

1| CREATE TABLE angajati
2|   (id_angajat number(5),
3|    nume varchar(25) constraint nume_ang_not_null not null,
4|    prenume varchar(25),
5|    email varchar(50),
6|    nr_telefon char(10) constraint nr_ang_not_null not null,
7|    data_angajare date default sysdate,
8|    tip_angajat varchar2(10) constraint tip_not_null not null,
9|    specializare varchar2(50),
10|    experienta number(2),
11|    CONSTRAINT angajati_pk PRIMARY KEY (id_angajat),
12|    unique(email),
13|    unique(nr_telefon)
14| );

```

Crearea tabelului *gătește_servește*:

```
1 CREATE TABLE gătește_servește
2   (id_angajat number(5) constraint ang_g_not_null not null,
3    id_comanda number(5) constraint comanda_g_not_null not null,
4    id_produs number(5) constraint produs_g_not_null not null,
5    cantitate number(2) constraint c_cantitate not null,
6    CONSTRAINT gătește_angajat_fk foreign key(id_angajat) references
7      angajati(id_angajat),
8    CONSTRAINT gătește_produs_fk foreign key(id_produs) references
9      produse(id_produs),
10   CONSTRAINT gătește_comanda_fk foreign key(id_comanda) references
11     comenzi(id_comanda),
12   CONSTRAINT pk_gătește primary key(id_angajat, id_comanda,id_produs),
13   check(cantitate>0)
14 );
```

5. Adăugarea informațiilor în tabele

Inserarea datelor în tabelul *clienti*:

```

1  --secventa pentru generarea codurilor clientilor
2  CREATE SEQUENCE SEQ_CLIENT
3    INCREMENT by 1
4    START WITH 1
5    MINVALUE 1
6    MAXVALUE 1000
7    NOCYCLE;
8
9    INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
10   VALUES (SEQ_CLIENT.NEXTVAL, 'Popescu', 'Maria', '0371692722');
11  INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
12   VALUES (SEQ_CLIENT.NEXTVAL, 'Radu', 'Andrei', '0772262672');
13  INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
14   VALUES (SEQ_CLIENT.NEXTVAL, 'Avram', 'Ana-Maria', '0727635283');
15  INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
16   VALUES (SEQ_CLIENT.NEXTVAL, 'Gavrilă', 'Cristina', '0742830745');
17  INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
18   VALUES (SEQ_CLIENT.NEXTVAL, 'Istrate', 'Bogdan-Ionut', '0739428220');
19  INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
20   VALUES (SEQ_CLIENT.NEXTVAL, 'Comnoiu', 'Adela-Ioana', '0762863289');
21  INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
22   VALUES (SEQ_CLIENT.NEXTVAL, 'Marin', 'Liviu', '0758263927');
23  INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
24   VALUES (SEQ_CLIENT.NEXTVAL, 'Marin', 'Liviu', '0758261234');
```

The screenshot shows the Oracle SQL Developer interface with a query window containing the following SQL statement:

```
SELECT * FROM clienti;
```

The results are displayed in a table titled "Query Result". The table has four columns: ID_CLIENT, NUME, PRENUME, and NR_TELEFON. The data is as follows:

ID_CLIENT	NUME	PRENUME	NR_TELEFON
1	Popescu	Maria	0371692722
2	Radu	Andrei	0772262672
3	Avram	Ana-Maria	0727635283
4	Gavrilă	Cristina	0742830745
5	Istrate	Bogdan-Ionut	0739428220
6	Comnoiu	Adela-Ioana	0762863289
7	Marin	Liviu	0758263927
8	Marin	Liviu	0758261234

Inserarea datelor în tabelul *tipuri_evenimente*:

```

1| INSERT INTO tipuri_evenimente
2|   VALUES (121, 'nunta', null);
3| INSERT INTO tipuri_evenimente
4|   VALUES (122, 'botez', null);
5| INSERT INTO tipuri_evenimente
6|   VALUES (123, 'aniversare', 'inclusiv majorat');
7| INSERT INTO tipuri_evenimente
8|   VALUES (124, 'revedere', 'liceu/facultate');
9| INSERT INTO tipuri_evenimente
10|    VALUES (125, 'cununie', null);

```

The screenshot shows the Oracle SQL Developer interface. A query window contains the command:

```
SELECT * FROM tipuri_evenimente;
```

The results are displayed in a table titled "Query Result". The table has three columns: ID_TIP_EVENTIMENT, DENUMIRE_TIP, and DESCRIERE. The data is as follows:

ID_TIP_EVENTIMENT	DENUMIRE_TIP	DESCRIERE
1	121 nunta	(null)
2	122 botez	(null)
3	123 aniversare	inclusiv majorat
4	124 revedere	liceu/facultate
5	125 cununie	(null)

Inserarea datelor în tabelul *evenimente*:

```

1| INSERT INTO evenimente
2|   VALUES (110, to_date('13-07-2021','dd-mm-yyyy'), 'decor albastru', 122);
3| INSERT INTO evenimente
4|   VALUES (111, to_date('25-06-2021','dd-mm-yyyy'), null, 125);
5| INSERT INTO evenimente
6|   VALUES (112, to_date('15-09-2020','dd-mm-yyyy'), null, 124);
7| INSERT INTO evenimente
8|   VALUES (113, to_date('03-04-2021','dd-mm-yyyy'), 'flori naturale', 121);
9| INSERT INTO evenimente
10|  VALUES (114, to_date('28-02-2021','dd-mm-yyyy'), 'decor rosu', 123);
11| INSERT INTO evenimente
12|  VALUES (115, to_date('28-02-2021','dd-mm-yyyy'), 'decor albastru', 124);
13| INSERT INTO evenimente
14|  VALUES (116, to_date('07-10-2021','dd-mm-yyyy'), null, 125);
15| INSERT INTO evenimente
16|  VALUES (117, to_date('12-02-2022','dd-mm-yyyy'), 'decor roz', 121);

```

SELECT * FROM evenimente;

Script Output x Query Result x

All Rows Fetched: 8 in 0.003 seconds

ID_EVENTAMENT	DATA_EVENTIMENT	PREFERINTE	ID_TIP_EVENTIMENT
1	110 13-JUL-21	decor albastru	122
2	111 25-JUN-21	(null)	125
3	112 15-SEP-20	(null)	124
4	113 03-APR-21	flori naturale	121
5	114 28-FEB-21	decor rosu	123
6	115 28-FEB-21	decor albastru	124
7	116 07-OCT-21	(null)	125
8	117 12-FEB-22	decor roz	121

Inserarea datelor în tabelul *moduri_de_plată*:

```

1| INSERT INTO moduri_de_plata
2|   VALUES (600, 'cash', null);
3| INSERT INTO moduri_de_plata
4|   VALUES (610, 'card', null);
5| INSERT INTO moduri_de_plata
6|   VALUES (620, 'online', null);
7| INSERT INTO moduri_de_plata
8|   VALUES (630, 'in rate', 'numai pentru evenimente');
9| INSERT INTO moduri_de_plata
10|    VALUES (640, 'tichet', null);

```

SELECT * FROM moduri_de_plata;

Script Output x Query Result x

All Rows Fetched: 5 in 0.008 seconds

ID_MOD_PLATA	DENUMIRE	OBSERVATII
1	600 cash	(null)
2	610 card	(null)
3	620 online	(null)
4	630 in rate	numai pentru evenimente
5	640 tichet	(null)

Inserarea datelor în tabelul *comenzi*:

```

1| --secvența pentru generarea codurilor comenzi
2| CREATE SEQUENCE SEQ_COMENZI
3| INCREMENT by 10
4| START WITH 10
5| MINVALUE 10
6| MAXVALUE 10000
7| NOCYCLE;
8|
9| INSERT INTO comenzi

```

```

10    VALUES(SEQ_COMENZI.NEXTVAL, to_date('09-03-2021','dd-mm-yyyy'), 600, 2,
11      113);
12  INSERT INTO comenzi
13    VALUES(SEQ_COMENZI.NEXTVAL, to_date('15-10-2020','dd-mm-yyyy'), 610, 5,
14      null);
15  INSERT INTO comenzi
16    VALUES(SEQ_COMENZI.NEXTVAL, to_date('03-05-2020','dd-mm-yyyy'), 620, 1,
17      null);
18  INSERT INTO comenzi
19    VALUES(SEQ_COMENZI.NEXTVAL, to_date('05-07-2021','dd-mm-yyyy'), 600, 3,
20      110);
21  INSERT INTO comenzi
22    VALUES(SEQ_COMENZI.NEXTVAL, to_date('18-10-2019','dd-mm-yyyy'), 630, 7,
23      null);
24  INSERT INTO comenzi
25    VALUES(SEQ_COMENZI.NEXTVAL, to_date('23-12-2020','dd-mm-yyyy'), 610, 4,
26      null);
27  INSERT INTO comenzi
28    VALUES(SEQ_COMENZI.NEXTVAL, to_date('16-02-2021','dd-mm-yyyy'), 620, 2,
      114);
  INSERT INTO comenzi
    VALUES(SEQ_COMENZI.NEXTVAL, to_date('30-04-2021','dd-mm-yyyy'), 600, 6,
      111);
  INSERT INTO comenzi
    VALUES(SEQ_COMENZI.NEXTVAL, to_date('02-08-2020','dd-mm-yyyy'), 630, 1,
      112);
  INSERT INTO comenzi
    VALUES(SEQ_COMENZI.NEXTVAL, to_date('05-10-2021','dd-mm-yyyy'), 600, 1,
      116);

```

SELECT * FROM comenzi;

Script Output x Query Result x

All Rows Fetched: 10 in 0.003 seconds

ID_COMANDA	DATA_COMANDA	ID_MOD_PLATA	ID_CLIENT	ID_EVENIMENT
1	10 09-MAR-21	600	2	113
2	20 15-OCT-20	610	5	(null)
3	30 03-MAY-20	620	1	(null)
4	40 05-JUL-21	600	3	110
5	50 18-OCT-19	630	7	(null)
6	60 23-DEC-20	610	4	(null)
7	70 16-FEB-21	620	2	114
8	80 30-APR-21	600	6	111
9	90 02-AUG-20	630	1	112
10	100 05-OCT-21	600	1	116

Inserarea datelor în tabelul *categorii*:

```

1 --secventa pentru generarea codurilor categoriilor
2 CREATE SEQUENCE SEQ_CATEGORII
3 INCREMENT by 10
4 START WITH 200
5 MINVALUE 200
6 MAXVALUE 10000
7 NOCYCLE;
8
9 INSERT INTO categorii
10     VALUES (SEQ_CATEGORII.NEXTVAL, 'paste');
11 INSERT INTO categorii
12     VALUES (SEQ_CATEGORII.NEXTVAL, 'salate');
13 INSERT INTO categorii
14     VALUES (SEQ_CATEGORII.NEXTVAL, 'pizza');
15 INSERT INTO categorii
16     VALUES (SEQ_CATEGORII.NEXTVAL, 'desert');
17 INSERT INTO categorii
18     VALUES (SEQ_CATEGORII.NEXTVAL, 'bauturi');
19 INSERT INTO categorii
20     VALUES (SEQ_CATEGORII.NEXTVAL, 'friptura');
21 INSERT INTO categorii
22     VALUES (SEQ_CATEGORII.NEXTVAL, 'garnituri');
```

The screenshot shows the Oracle SQL Developer interface. In the top-left, there is a code editor window containing the SQL command: `SELECT * FROM categorii;`. Below it is a toolbar with icons for Script Output, Query Result, and SQL. The status bar indicates "All Rows Fetched: 7 in 0.003 seconds". The main area displays a grid of data from the `categorii` table:

ID_CATEGORIE	NUME_CATEGORIE
1	200 paste
2	210 salate
3	220 pizza
4	230 desert
5	240 bauturi
6	250 friptura
7	260 garnituri

Inserarea datelor în tabelul *furnizori*:

```

1 --secventa pentru generarea codurilor furnizorilor
2 CREATE SEQUENCE SEQ_FURNIZOR
3 INCREMENT by 10
4 START WITH 1000
5 MINVALUE 1000
6 MAXVALUE 90000
7 NOCYCLE;
8
9 INSERT INTO furnizori
```

```

10     VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC Bucuria Gustului', '0764652224');
11 INSERT INTO furnizori
12     VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC Gustos din Natura', '0775375472');
13 INSERT INTO furnizori
14     VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC CrisFoods', '0775264221');
15 INSERT INTO furnizori
16     VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC De Bun Gust', '0787625242');
17 INSERT INTO furnizori
18     VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC Delicios', '0769727366');
19 INSERT INTO furnizori
20     VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC Natural', '0769721234');
21 INSERT INTO furnizori
22     VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC Natural', '0769712345');

```

SELECT * FROM furnizori;

ID_FURNIZOR	DENUMIRE	NR_TELEFON
1	1000 SC Bucuria Gustului	0764652224
2	1010 SC Gustos din Natura	0775375472
3	1020 SC CrisFoods	0775264221
4	1030 SC De Bun Gust	0787625242
5	1040 SC Delicios	0769727366
6	1050 SC Natural	0769721234
7	1060 SC Natural	0769712345

Inserarea datelor în tabelul *ingredient*:

```

1 --secrenta pentru generarea codurilor ingredientelor
2 CREATE SEQUENCE SEQ_INGREDIENT
3 INCREMENT by 1
4 START WITH 170
5 MINVALUE 170
6 MAXVALUE 90000
7 NOCYCLE;
8
9 INSERT INTO ingrediente
10    VALUES(SEQ_INGREDIENT.NEXTVAL, 'spaghete', 1000);
11 INSERT INTO ingrediente
12    VALUES(SEQ_INGREDIENT.NEXTVAL, 'sos de rosii', 1000);
13 INSERT INTO ingrediente
14    VALUES(SEQ_INGREDIENT.NEXTVAL, 'carne tocata', 1010);
15 INSERT INTO ingrediente
16    VALUES(SEQ_INGREDIENT.NEXTVAL, 'parmezan', 1020);
17 INSERT INTO ingrediente
18    VALUES(SEQ_INGREDIENT.NEXTVAL, 'salata verde', 1030);
19 INSERT INTO ingrediente

```

```

20     VALUES(SEQ_INGREDIENT.NEXTVAL, 'piept de pui', 1020);
21 INSERT INTO ingrediente
22     VALUES(SEQ_INGREDIENT.NEXTVAL, 'rosii', 1000);
23 INSERT INTO ingrediente
24     VALUES(SEQ_INGREDIENT.NEXTVAL, 'castraveti', 1040);
25 INSERT INTO ingrediente
26     VALUES(SEQ_INGREDIENT.NEXTVAL, 'branza', 1040);
27 INSERT INTO ingrediente
28     VALUES(SEQ_INGREDIENT.NEXTVAL, 'faina', 1020);
29 INSERT INTO ingrediente
30     VALUES(SEQ_INGREDIENT.NEXTVAL, 'masline', 1030);
31 INSERT INTO ingrediente
32     VALUES(SEQ_INGREDIENT.NEXTVAL, 'piscoturi', 1000);
33 INSERT INTO ingrediente
34     VALUES(SEQ_INGREDIENT.NEXTVAL, 'mascarpone', 1010);
35 INSERT INTO ingrediente
36     VALUES(SEQ_INGREDIENT.NEXTVAL, 'ciocolata', 1010);
37 INSERT INTO ingrediente
38     VALUES(SEQ_INGREDIENT.NEXTVAL, 'apa', 1030);
39 INSERT INTO ingrediente
40     VALUES(SEQ_INGREDIENT.NEXTVAL, 'suc', 1040);
41 INSERT INTO ingrediente
42     VALUES(SEQ_INGREDIENT.NEXTVAL, 'vin', 1010);
43 INSERT INTO ingrediente
44     VALUES(SEQ_INGREDIENT.NEXTVAL, 'carne porc', 1020);
45 INSERT INTO ingrediente
46     VALUES(SEQ_INGREDIENT.NEXTVAL, 'carne pui', 1020);

```

SELECT * FROM ingrediente;

ID_INGREDIENT	DENUMIRE	ID_FURNIZOR
1	170 spaghetti	1000
2	171 sos de rosii	1000
3	172 carne tocata	1010
4	173 parmezan	1020
5	174 salata verde	1030
6	175 piept de pui	1020
7	176 rosii	1000
8	177 castraveti	1040
9	178 branza	1040
10	179 faina	1020
11	180 masline	1030
12	181 piscoturi	1000
13	182 mascarpone	1010
14	183 ciocolata	1010
15	184 apa	1030
16	185 suc	1040
17	186 vin	1010
18	187 carne porc	1020
19	188 carne pui	1020

Inserarea datelor în tabelul *produse*:

```

1  --secreta pentru generarea codurilor produselor
2  CREATE SEQUENCE SEQ_PRODUS
3  INCREMENT by 1
4  START WITH 50
5  MINVALUE 50
6  MAXVALUE 90000
7  NOCYCLE;
8
9  INSERT INTO produse
10    VALUES(SEQ_PRODUS.NEXTVAL, 'paste carbonara', 24, '300g', 200);
11  INSERT INTO produse
12    VALUES(SEQ_PRODUS.NEXTVAL, 'paste bolognese', 22, null, 200);
13  INSERT INTO produse
14    VALUES(SEQ_PRODUS.NEXTVAL, 'salata cu pui', 23, '200g', 210);
15  INSERT INTO produse
16    VALUES(SEQ_PRODUS.NEXTVAL, 'salata greceasca', 16, '200g', 210);
17  INSERT INTO produse
18    VALUES(SEQ_PRODUS.NEXTVAL, 'pizza margherita', 17, 'medie', 220);
19  INSERT INTO produse
20    VALUES(SEQ_PRODUS.NEXTVAL, 'pizza capriciosa', 19, 'medie', 220);
21  INSERT INTO produse
22    VALUES(SEQ_PRODUS.NEXTVAL, 'tiramisu', 15, null, 230);
23  INSERT INTO produse
24    VALUES(SEQ_PRODUS.NEXTVAL, 'clatite', 14, null, 230);
25  INSERT INTO produse
26    VALUES(SEQ_PRODUS.NEXTVAL, 'apa', 5, '500ml', 240);
27  INSERT INTO produse
28    VALUES(SEQ_PRODUS.NEXTVAL, 'suc', 7, 'diverse sortimente', 240);
29  INSERT INTO produse
30    VALUES(SEQ_PRODUS.NEXTVAL, 'vin', 10, 'la sticla', 240);
31  INSERT INTO produse
32    VALUES(SEQ_PRODUS.NEXTVAL, 'friptura pui', 25, '230g', 250);
33  INSERT INTO produse
34    VALUES(SEQ_PRODUS.NEXTVAL, 'friptura porc', 28, '200g', 250);

```

The screenshot shows a MySQL Workbench interface. In the top query editor, the SQL command is:

```
SELECT * FROM produse;
```

In the bottom pane, the results are displayed in a table titled "Query Result". The table has four columns: ID_PRODUS, DENUMIRE, PRET, and OBSERVATII. The data consists of 13 rows, each representing a product with its name, price, and a note about observations.

ID_PRODUS	DENUMIRE	PRET	OBSERVATII	ID_CATEGORIE
1	50 paste carbonara	24 300g		200
2	51 paste bolognese	22 (null)		200
3	52 salata cu pui	23 200g		210
4	53 salata greceasca	16 200g		210
5	54 pizza margherita	17 medie		220
6	55 pizza capriciosa	19 medie		220
7	56 tiramisu	15 (null)		230
8	57 clatite	14 (null)		230
9	58 apa	5 500ml		240
10	59 suc	7 diverse sortimente		240
11	60 vin	10 la sticla		240
12	61 friptura pui	25 230g		250
13	62 friptura porc	28 200g		250

Inserarea datelor în tabelul *contine*:

```

1  INSERT INTO contine
2      VALUES(50,170);
3  INSERT INTO contine
4      VALUES(50,173);
5  INSERT INTO contine
6      VALUES(51,170);
7  INSERT INTO contine
8      VALUES(51,171);
9  INSERT INTO contine
10     VALUES(51,172);
11  INSERT INTO contine
12     VALUES(52,174);
13  INSERT INTO contine
14     VALUES(52,175);
15  INSERT INTO contine
16     VALUES(53,174);
17  INSERT INTO contine
18     VALUES(53,176);
19  INSERT INTO contine
20     VALUES(53,177);
21  INSERT INTO contine
22     VALUES(53,178);
23  INSERT INTO contine
24     VALUES(54,179);
25  INSERT INTO contine
26     VALUES(54,171);
27  INSERT INTO contine
28     VALUES(54,173);
29  INSERT INTO contine
30     VALUES(55,179);
```

```
31| INSERT INTO contine
32|     VALUES(55,180);
33| INSERT INTO contine
34|     VALUES(55,171);
35| INSERT INTO contine
36|     VALUES(55,173);
37| INSERT INTO contine
38|     VALUES(56,181);
39| INSERT INTO contine
40|     VALUES(56,182);
41| INSERT INTO contine
42|     VALUES(57,179);
43| INSERT INTO contine
44|     VALUES(57,183);
45| INSERT INTO contine
46|     VALUES(58,184);
47| INSERT INTO contine
48|     VALUES(59,185);
49| INSERT INTO contine
50|     VALUES(60,186);
```

| SELECT * FROM contine;

Script Output x Query Result x

SQL | All Rows Fetched: 25 in 0.004 seconds

ID_PRODUS	ID_INGREDIENT
1	50
2	50
3	51
4	51
5	51
6	52
7	52
8	53
9	53
10	53
11	53
12	54
13	54
14	54
15	55
16	55
17	55
18	55
19	56
20	56
21	57
22	57
23	58
24	59
25	60

Inserarea datelor în tabelul *angajati*:

```

1  INSERT INTO angajati
2      VALUES(200, 'Avram', 'Alina', 'alina@angajat.ro', '0765242562',
3             to_date('03-03-2018','dd-mm-yyyy'), 'bucatar', 'Franta', null);
4  INSERT INTO angajati
5      VALUES(201, 'Dumitrache', 'Bianca-Maria', 'bianca@angajat.ro',
6             '0775252425', to_date('15-10-2019','dd-mm-yyyy'), 'ospatar', null, 2);
7  INSERT INTO angajati
8      VALUES(202, 'Vilcu', 'Elena', 'ella@angajat.ro', '0738726252',
9             to_date('19-07-2019','dd-mm-yyyy'), 'bucatar', 'Romania', null);
10 INSERT INTO angajati
11     VALUES(203, 'Nistor', 'Izabela', 'iza@angajat.ro', '0758625279',
12            to_date('07-02-2021','dd-mm-yyyy'), 'ospatar', null, 3);
13 INSERT INTO angajati
14     VALUES(204, 'Dinu', 'Marian', 'marian@angajat.ro', '0787534299',
15            to_date('21-05-2015','dd-mm-yyyy'), 'bucatar', 'Spania', null);
16 INSERT INTO angajati
17     VALUES(205, 'Ionsecu', 'Ion', 'ion@angajat.ro', '0749293633',
18            to_date('18-04-2020','dd-mm-yyyy'), 'ospatar', null, null);
19 INSERT INTO angajati
20     VALUES(206, 'Brezan', 'Sabin-Alexandru', 'sabin@angajat.ro', '0758622885',
21            to_date('11-09-2020','dd-mm-yyyy'), 'bucatar', 'Franta', null);
22 INSERT INTO angajati
23     VALUES(207, 'Albu', 'Georgiana', 'geo@angajat.ro', '0769628434',
24            to_date('13-01-2021','dd-mm-yyyy'), 'ospatar', null, 6);
25 INSERT INTO angajati
26     VALUES(208, 'Brezan', 'Sabin-Alexandru', 'sabin2@angajat.ro', '0723961024',
27            to_date('03-04-2021','dd-mm-yyyy'), 'bucatar', null, null);
28 INSERT INTO angajati
29     VALUES(209, 'Stroilescu', 'Maria', 'maria@angajat.ro', '0722567024',
30            to_date('23-09-2021','dd-mm-yyyy'), 'bucatar', null, null);

```

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

```
SELECT * FROM angajati;
```

The results are displayed in a table titled "Query Result". The table has 10 rows and 9 columns. The columns are labeled: ID_ANGAJAT, NUME, PRENUME, EMAIL, NR_TELEFON, DATA_ANGAJARE, TIP_ANGAJAT, SPECIALIZARE, and EXPERIENTA.

ID_ANGAJAT	NUME	PRENUME	EMAIL	NR_TELEFON	DATA_ANGAJARE	TIP_ANGAJAT	SPECIALIZARE	EXPERIENTA	
1	200	Avram	Alina	alina@angajat.ro	0765242562	03-MAR-18	bucatar	Franta	(null)
2	201	Dumitrache	Bianca-Maria	bianca@angajat.ro	0775252425	15-OCT-19	ospatar	(null)	2
3	202	Vilcu	Elena	ella@angajat.ro	0738726252	19-JUL-19	bucatar	Romania	(null)
4	203	Nistor	Izabela	iza@angajat.ro	0758625279	07-FEB-21	ospatar	(null)	3
5	204	Dinu	Marian	marian@angajat.ro	0787534299	21-MAY-15	bucatar	Spania	(null)
6	205	Ionsecu	Ion	ion@angajat.ro	0749293633	18-APR-20	ospatar	(null)	(null)
7	206	Brezan	Sabin-Alexandru	sabin@angajat.ro	0758622885	11-SEP-20	bucatar	Franta	(null)
8	207	Albu	Georgiana	geo@angajat.ro	0769628434	13-JAN-21	ospatar	(null)	6
9	208	Brezan	Sabin-Alexandru	sabin2@angajat.ro	0723961024	03-APR-21	bucatar	(null)	(null)
10	209	Stroilescu	Maria	maria@angajat.ro	0722567024	23-SEP-21	bucatar	(null)	(null)

Inserarea datelor în tabelul *gătește_servește*:

```
1  INSERT INTO gătește_servește
2      VALUES(204,10,62,20);
3  INSERT INTO gătește_servește
4      VALUES(206,10,56,15);
5  INSERT INTO gătește_servește
6      VALUES(207,10,59,20);
7  INSERT INTO gătește_servește
8      VALUES(200,20,55,2);
9  INSERT INTO gătește_servește
10     VALUES(202,30,57,4);
11 INSERT INTO gătește_servește
12     VALUES(205,30,59,4);
13 INSERT INTO gătește_servește
14     VALUES(204,40,61,10);
15 INSERT INTO gătește_servește
16     VALUES(203,40,60,8);
17 INSERT INTO gătește_servește
18     VALUES(200,50,51,3);
19 INSERT INTO gătește_servește
20     VALUES(206,60,53,6);
21 INSERT INTO gătește_servește
22     VALUES(204,70,55,16);
23 INSERT INTO gătește_servește
24     VALUES(207,70,59,16);
25 INSERT INTO gătește_servește
26     VALUES(206,80,62,22);
27 INSERT INTO gătește_servește
28     VALUES(201,80,60,22);
29 INSERT INTO gătește_servește
30     VALUES(202,90,50,5);
31 INSERT INTO gătește_servește
32     VALUES(201,90,58,7);
33 INSERT INTO gătește_servește
34     VALUES(207,10,62,20);
35 INSERT INTO gătește_servește
36     VALUES(207,10,56,15);
37 INSERT INTO gătește_servește
38     VALUES(201,20,55,2);
39 INSERT INTO gătește_servește
40     VALUES(205,30,57,4);
41 INSERT INTO gătește_servește
42     VALUES(203,40,61,10);
43 INSERT INTO gătește_servește
44     VALUES(201,50,51,3);
45 INSERT INTO gătește_servește
46     VALUES(205,60,53,6);
```

```
47| INSERT INTO gateste_serveste
48|     VALUES(207,70,55,16);
49| INSERT INTO gateste_serveste
50|     VALUES(203,80,62,22);
51| INSERT INTO gateste_serveste
52|     VALUES(205,90,50,5);
53| INSERT INTO gateste_serveste
54|     VALUES(203,10,58,3);
55| INSERT INTO gateste_serveste
56|     VALUES(204,100,52,7);
57| INSERT INTO gateste_serveste
58|     VALUES(205,100,52,7);
```

SELECT * FROM gateste_serveste;

Script Output x Query Result x
SQL | All Rows Fetched: 29 in 0.003 seconds

ID_ANAJAT	ID_COMANDA	ID_PRODUS	CANTITATE
1	204	10	62
2	206	10	56
3	207	10	59
4	200	20	55
5	202	30	57
6	205	30	59
7	204	40	61
8	203	40	60
9	200	50	51
10	206	60	53
11	204	70	55
12	207	70	59
13	206	80	62
14	201	80	60
15	202	90	50
16	201	90	58
17	207	10	62
18	207	10	56
19	201	20	55
20	205	30	57
21	203	40	61
22	201	50	51
23	205	60	53
24	207	70	55
25	203	80	62
26	205	90	50
27	203	10	58
28	204	100	52
29	205	100	52

6. Subprogram stocat care utilizează două tipuri de colecție

Definiți o funcție care primește ca parametru denumirea unui furnizor și returnează numărul de comenzi care contin cel puțin un produs gătit cu ingrediente de la furnizorul dat.

În rezolvarea exercitiului am folosit:

- un vector cu dimensiune maxima 100 pentru a reține ingredientele furnizate de furnizorul dat
 - un tablou indexat pentru a reține lista de produse ce conțin ingrediente din vector
 - un tablou imbricat pentru a retine comenziile ce contin produse din tabloul indexat

La final funcția a returnat numărul de elemente din tabloul imbricat (numărul de comenzi ce îndeplinesc condiția).

```

1 CREATE OR REPLACE FUNCTION comenzi_furnizor
2   (nume furnizori.denumire%TYPE)
3 RETURN NUMBER IS
4   TYPE vector IS VARRAY(100) OF ingrediente.id_ingredient%TYPE; --presupunem
5     ca nu avem mai mult de 100 de ingrediente
6   t_ing vector := vector(); --vector folosit pentru a retine lista de
7     ingrediente furnizate
8
9   TYPE tablou_indexat IS TABLE OF produse.id_produs%TYPE INDEX BY PLS_INTEGER;
10  t_prod tablou_indexat; --tablou indexat folosit pentru a retine lista de
11    produse ce contin ingredientele furnizorului
12  t_prod_aux tablou_indexat; --tablou indexat auxiliar folosit pentru a adauga
13    in tabloul indexat mai sus definit numai elemente distincte
14
15  TYPE tablou_imbricat IS TABLE OF comenzi.id_comanda%TYPE;
16  t_comenzi tablou_imbricat:= tablou_imbricat(); --tablou imbricat folosit
17    pentru a retine comenzile ce contin produse din t_prod
18  t_comenzi_aux tablou_imbricat:= tablou_imbricat();--tablou imbricat auxiliar
19    folosit pentru a adauga in tabloul imbricat mai sus definit numai
      elemente distincte

20
21  id_ing ingrediente.id_ingredient%TYPE;
22  id_prod produse.id_produs%TYPE;
23  v_id furnizori.id_furnizor%TYPE;
24  aux NUMBER;

```

```

20|| BEGIN
21|   SELECT id_furnizor --selectez id-ul furnizorului dat
22|     INTO v_id
23|   FROM furnizori
24| WHERE upper(denumire) = upper(nume);
25|
26|   SELECT id_ingredient --introduc in vectorul t_ing id-urile ingredientelor
27|     BULK COLLECT INTO t_ing
28|   FROM ingrediente
29| WHERE id_furnizor = v_id;
30|
31|   DBMS_OUTPUT.PUT_LINE('Id furnizor: ' || v_id);
32|   DBMS_OUTPUT.PUT_LINE('Ingrediente: ');
33|   FOR i IN t_ing.FIRST..t_ing.LAST LOOP --afisare id-uri ingrediente
34|     DBMS_OUTPUT.PUT_LINE(t_ing(i));
35|   END LOOP;
36|
37|   id_ing := t_ing(t_ing.FIRST);
38|
39|   SELECT DISTINCT id_produs --initializez tabloul indexat cu produsele gatite
40|     cu primul ingredient
41|   BULK COLLECT INTO t_prod
42|   FROM contine
43| WHERE id_ingredient = id_ing;
44|
45|   FOR i IN t_ing.FIRST+1..t_ing.LAST LOOP -- pentru fiecare ingredient ramas
46|
47|     id_ing := t_ing(i);
48|     SELECT DISTINCT id_produs --introduc in tabloul indexat auxiliar
49|       produsele gatite cu ingredientul curent
50|     BULK COLLECT INTO t_prod_aux
51|     FROM contine
52|     WHERE id_ingredient = id_ing;
53|
54|     --verificare daca exista deja produsul in lista finala
55|     FOR k IN t_prod_aux.FIRST..t_prod_aux.LAST LOOP
56|       aux := 0;
57|
58|       FOR j IN t_prod.FIRST..t_prod.LAST LOOP
59|         IF t_prod_aux(k) = t_prod(j) THEN
60|           aux := 1;
61|         END IF;
62|       END LOOP;
63|
64|       --daca aux = 0 inseamna ca produsul nu exista deja in lista, asa ca
65|         il adaugam la final
66|       IF aux = 0 THEN

```

```

64      t_prod(t_prod.LAST + 1) := t_prod_aux(k);
65  END IF;
66
67  END LOOP;
68 END LOOP;
69
70 DBMS_OUTPUT.PUT_LINE('Produse: ');
71 FOR i IN t_prod.FIRST..t_prod.LAST LOOP --afisare id-uri produse
72   DBMS_OUTPUT.PUT_LINE(t_prod(i));
73 END LOOP;
74
75 id_prod := t_prod(t_prod.FIRST);
76 SELECT DISTINCT id_comanda --initializez tabloul imbricat t_comenzi cu id-ul
77   comenzilor ce contin primul produs din t_prod
78 BULK COLLECT INTO t_comenzi
79 FROM gateste_serveste
80 WHERE id_produs = id_prod;
81
82 FOR i IN t_prod.FIRST+1..t_prod.LAST LOOP
83   id_prod := t_prod(i);
84   SELECT DISTINCT id_comanda --introduc in tabloul imbricat auxiliar id-ul
85     comenzilor ce contin primul produs din t_prod
86   BULK COLLECT INTO t_comenzi_aux
87   FROM gateste_serveste
88   WHERE id_produs = id_prod;
89
90   --verificare daca exista deja comanda in lista finala
91   FOR k IN t_comenzi_aux.FIRST..t_comenzi_aux.LAST LOOP
92     aux := 0;
93
94     FOR j IN t_comenzi.FIRST..t_comenzi.LAST LOOP
95       IF t_comenzi_aux(k) = t_comenzi(j) THEN
96         aux := 1;
97       END IF;
98     END LOOP;
99
100    --daca aux = 0 inseamna ca id-ul comenzii nu exista deja in lista,
101    --asa ca il adaugam la final
102    IF aux = 0 THEN
103      t_comenzi.EXTEND;
104      t_comenzi(t_comenzi.LAST) := t_comenzi_aux(k);
105    END IF;
106  END LOOP;
107 END LOOP;
108
109 DBMS_OUTPUT.PUT_LINE('Comenzi: ');
110 FOR i IN t_comenzi.FIRST..t_comenzi.LAST LOOP --afisare id-uri comenzi
111

```

```

108     DBMS_OUTPUT.PUT_LINE(t_comenzi(i));
109   END LOOP;
110
111   RETURN t_comenzi.COUNT; --returnare numar de elemente ale tabloului imbricat
112   cu id-urile comenzilor
113
114   EXCEPTION
115     WHEN NO_DATA_FOUND THEN --cazul in care denumirea data nu corespunde cu
116       denumirea unui furnizor
117       RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun furnizor cu
118       denumirea data');
119     WHEN TOO_MANY_ROWS THEN --cazul in care exista mai multi furnizori cu
120       denumirea introdusa
121       RAISE_APPLICATION_ERROR(-20001, 'Există mai mulți furnizori cu
122       denumirea data');
123   END comenzi_furnizor;
124 /
125
126 BEGIN
127   DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor('SC Gustos
128   din Natura'));
129   --DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor('SC Alt
130   Nume')); --exceptie: no_data_found
131   --DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor('SC
132   Natural'))); --exceptie: to_many_rows
133 END;
134 /
135
136 --pentru a introduce denumirea de la tastatura
137
138 DECLARE
139   nume furnizori.denumire%TYPE := '&p_nume';
140
141 BEGIN
142   DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor(nume));
143 END;
144 /

```

Exemplu apelare funcție cu denumirea unui furnizor care există și este unică:

```

projectSGBD_create.sql projectSGBD_6-9.sql projectSGBD_10-14.sql projectSGBD_ex14.sql
SQL Worksheet History
projectSGBD
Worksheet Query Builder
DBMS_OUTPUT.PUT_LINE('Comenzi:');
FOR i in t_comenzi.FIRST..t_comenzi.LAST LOOP --afisare id-uri comenzi
    DBMS_OUTPUT.PUT_LINE(t_comenzi(i));
END LOOP;

RETURN t_comenzi.COUNT; --returnare numar de elemente ale tabloului imbricat cu id-urile comenziilor

EXCEPTION
    WHEN NO_DATA_FOUND THEN --cazul in care denumirea data nu corespunde cu denumirea unui furnizor
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun furnizor cu denumirea data');
    WHEN TOO_MANY_ROWS THEN --cazul in care exista mai multi furnizori cu denumirea introdusa
        RAISE_APPLICATION_ERROR(-20001, 'Există mai mulți furnizori cu denumirea data');
END comenzi_furnizor;
/

BEGIN
    DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor('SC Gustos din Natura'));
    --DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor('SC Alt Nume')); --exceptie: no_data_found
    --DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor('SC Natural')) --exceptie: too_many_rows
END;
/
--pentru a introduce denumirea de la tastatura
DECLARE
    nume furnizori.denumire%TYPE := '&p_nume';
BEGIN
    DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor(nume));
END;

```

Query Result | Script Output | Task completed in 0.068 seconds

ORA-06512: at line 4

PL/SQL procedure successfully completed.

Exemplu apelare funcție cu denumirea unui furnizor care nu există:

```

projectSGBD_create.sql projectSGBD_6-9.sql projectSGBD_10-14.sql projectSGBD_ex14.sql
SQL Worksheet History
projectSGBD
Worksheet Query Builder
DBMS_OUTPUT.PUT_LINE('Comenzi:');
FOR i in t_comenzi.FIRST..t_comenzi.LAST LOOP --afisare id-uri comenzi
    DBMS_OUTPUT.PUT_LINE(t_comenzi(i));
END LOOP;

RETURN t_comenzi.COUNT; --returnare numar de elemente ale tabloului imbricat cu id-urile comenziilor

EXCEPTION
    WHEN NO_DATA_FOUND THEN --cazul in care denumirea data nu corespunde cu denumirea unui furnizor
        RAISE_APPLICATION_ERROR(-20000, 'Nu există niciun furnizor cu denumirea data');
    WHEN TOO_MANY_ROWS THEN --cazul in care există mai mulți furnizori cu denumirea introdusă
        RAISE_APPLICATION_ERROR(-20001, 'Există mai mulți furnizori cu denumirea data');
END comenzi_furnizor;
/

BEGIN
    --DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor('SC Gustos din Natura'));
    DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor('SC Alt Nume')); --exceptie: no_data_found
    --DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor('SC Natural')) --exceptie: too_many_rows
END;
/
--pentru a introduce denumirea de la tastatura

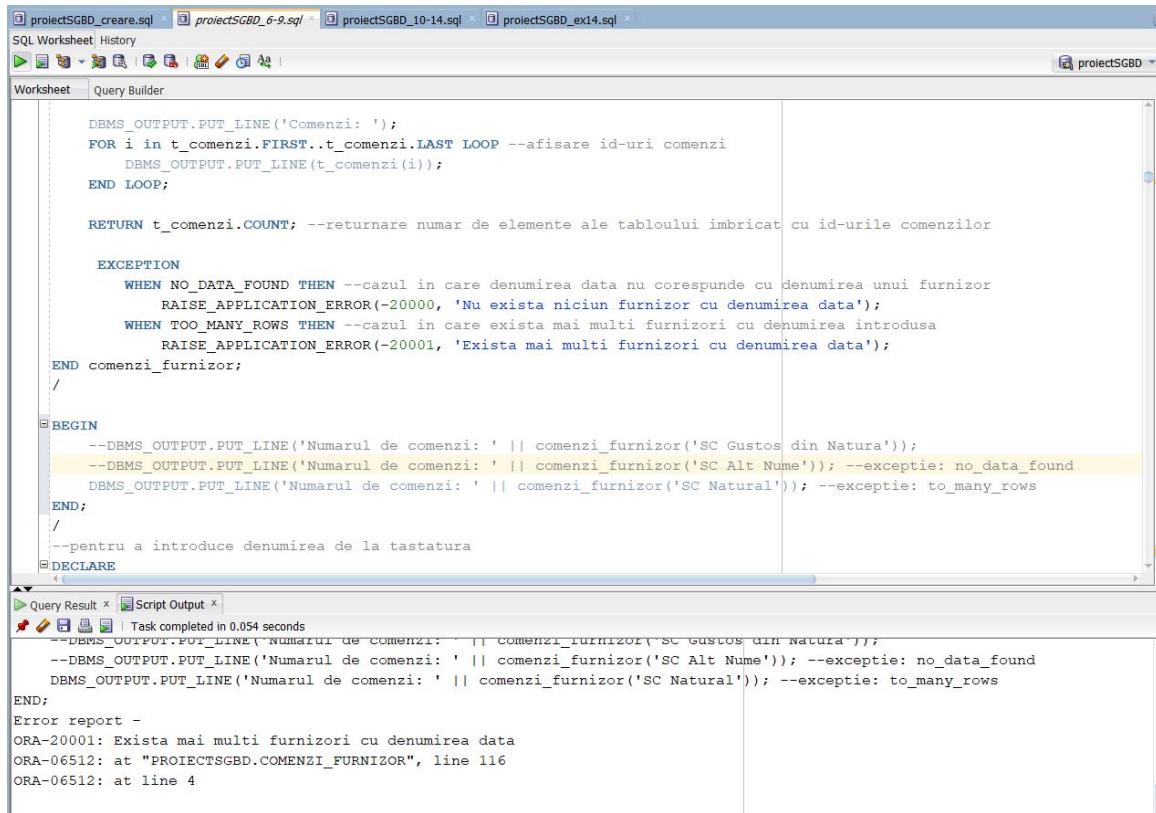
```

Query Result | Script Output | Task completed in 0.07 seconds

Error report -

ORA-20000: Nu există niciun furnizor cu denumirea data
ORA-06512: at "PROIECTSGBD.COMENZI_FURNIZOR", line 114
ORA-06512: at line 3
20000. 00000 - "%"
*Cause: The stored procedure 'raise_application_error'
was called which causes this error to be generated.
*Action: Correct the problem as described in the error message or contact
the application administrator or DBA for more information.

Exemplu apelare funcție cu o denumire care corespunde pentru mai mulți furnizori:



The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, there are tabs for 'projectSGBD_reate.sql', 'projectSGBD_6-9.sql' (which is currently selected), 'projectSGBD_10-14.sql', and 'projectSGBD_ex14.sql'. Below the tabs, the title bar says 'SQL Worksheet History' and 'projectSGBD'. The main area is divided into two panes: 'Worksheet' and 'Query Builder'. The 'Worksheet' pane contains a PL/SQL block. The 'Query Result' pane at the bottom shows the execution results and errors.

```

DBMS_OUTPUT.PUT_LINE('Comenzi: ');
FOR i IN t_comenzi.FIRST..t_comenzi.LAST LOOP --afisare id-uri comenzi
    DBMS_OUTPUT.PUT_LINE(t_comenzi(i));
END LOOP;

RETURN t_comenzi.COUNT; --returnare numar de elemente ale tabloului imbucat cu id-urile comenziilor

EXCEPTION
    WHEN NO_DATA_FOUND THEN --cazul in care denumirea data nu corespunde cu denumirea unui furnizor
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun furnizor cu denumirea data');
    WHEN TOO_MANY_ROWS THEN --cazul in care exista mai multi furnizori cu denumirea introdusa
        RAISE_APPLICATION_ERROR(-20001, 'Există mai mulți furnizori cu denumirea data');
END comenzi_furnizor;
/

BEGIN
    --DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor('SC Gustos din Natura'));
    --DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor('SC Alt Nume')); --exceptie: no_data_found
    DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor('SC Natural')) --exceptie: to_many_rows
END;
/
--pentru a introduce denumirea de la tastatura
DECLARE

```

Query Result pane content:

```

Task completed in 0.054 seconds
--DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor('SC Gustos din Natura'));
--DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor('SC Alt Nume')); --exceptie: no_data_found
DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || comenzi_furnizor('SC Natural')) --exceptie: to_many_rows
END;
Error report -
ORA-20001: Există mai mulți furnizori cu denumirea data
ORA-06512: at "PROIECTSGBD.COMENZI_FURNIZOR", line 116
ORA-06512: at line 4

```

7. Subprogram stocat care utilizează un cursor

Să se scrie o procedură care pentru numele unui bucătar să afișeze produsele pe care le-a gătit acesta (o singură dată fiecare), iar pentru fiecare produs să afișeze și ingredientele folosite.

Pentru a rezolva problema am utilizat un cursor clasic pentru a stoca și procesa produsele gătite de bucătar și un ciclu cursor parametrizat pentru a stoca și parcurge ingredientele pentru fiecare produs.

```

1  CREATE OR REPLACE PROCEDURE produse_gatite
2      (v_nume angajati.nume%TYPE,
3       v_prenume angajati.prenume%TYPE)
4  IS
5      v_id angajati.id_angajat%TYPE;
6      v_prod produse.id_produs%TYPE;
7      nume_prod produse.denumire%TYPE;
8      --cursor clasic
9      CURSOR c_prod IS
10         SELECT DISTINCT id_produs
11            FROM gateste_serveste
12           WHERE id_angajat = (SELECT id_angajat
13                           FROM angajati
14                          WHERE upper(nume)=upper(v_nume) AND
15                                upper(prenume)=upper(v_prenume)
16                                AND tip_angajat = 'bucatar'
17                                );
18      --ciclul cursor parametrizat
19      CURSOR c_ingr(id_prod produse.id_produs%TYPE) IS
20          SELECT denumire
21            FROM ingrediente
22           WHERE id_ingredient IN (SELECT id_ingredient
23                           FROM contine
24                             WHERE id_produs = id_prod
25                             );
26      BEGIN
27          SELECT id_angajat --selectare id bucatar
28             INTO v_id
29             FROM angajati
30             WHERE upper(nume)=upper(v_nume) AND upper(prenume)=upper(v_prenume) AND
31                   tip_angajat = 'bucatar';

```

```

30 --am pus si conditia ca numele introdus sa fie al unui bucatar
31
32 DBMS_OUTPUT.PUT_LINE('BUCATARUL ' || initcap(v_nume) || ', ' || 
33   initcap(v_prenume));
34
35 OPEN c_prod; --deschidere cursor
36 LOOP FETCH c_prod INTO v_prod; --incarcare linii pe rand in variabila
37   v_prod
38 EXIT WHEN c_prod%NOTFOUND;
39   SELECT denumire --selectare denumire produs
40     INTO nume_prod
41     FROM produse
42     WHERE id_produs = v_prod;
43   DBMS_OUTPUT.PUT_LINE('Produsul: ' || nume_prod);
44   FOR ingr IN c_ingr(v_prod) LOOP --ciclu cursor pentru afisarea
45     ingredientelor
46       DBMS_OUTPUT.PUT_LINE(' -> ' || ingr.denumire);
47   END LOOP;
48   DBMS_OUTPUT.NEW_LINE;
49 END LOOP;
50
51 --daca numarul liniilor este 0, atunci inseamna ca bucatarul nu a gatit
52   nimic inca
53 IF c_prod%ROWCOUNT = 0 THEN
54   DBMS_OUTPUT.PUT_LINE('Bucatarul dat nu a gatit niciun produs pana
55     acum');
56 END IF;
57 CLOSE c_prod; --inchidere cursor
58
59 EXCEPTION
60   WHEN NO_DATA_FOUND THEN --cazul in care nu exista niciun bucatar cu
61     numele dat
62     RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun bucatar cu
63       numele dat');
64   WHEN TOO_MANY_ROWS THEN --cazul in care exista mai multi bucatari cu
65     numele dat
66     RAISE_APPLICATION_ERROR(-20001, 'Există mai mulți bucătari cu
67       numele dat');
68 END produse_gatite;
69 /
70
71 BEGIN
72   produse_gatite('Avram', 'Alina');
73   --produse_gatite('Ana', 'Maria'); -- exceptie no_data_found (nu exista
74     niciun bucatar cu acest nume)
75   --produse_gatite('Brezan', 'Sabin-Alexandru'); -- exceptie too_many_rows
76     (există doi bucatari cu acest nume)

```

```

66      produse_gatite('Stroilescu', 'Maria'); -- Bucatarul dat nu a gatit niciun
67          produs pana acum
68
69      END;
70  /
71  --pentru a introduce numele bucatarului de la tastatura
72  DECLARE
73      nume angajati.nume%TYPE := '&p_nume';
74      prenume angajati.prenume%TYPE := '&p_prenume';
75  BEGIN
76      produse_gatite(nume, prenume);
77  END;
78  /

```

Exemplu apelare procedură pentru un bucătar care există, are un nume unic și a gătit produse până acum:

The screenshot shows the Oracle SQL Developer interface with a PL/SQL script being run. The script calls the `produse_gatite` procedure with parameters 'Avram' and 'Alina'. The output window displays the results of the procedure execution, showing the bucătar's name and the list of dishes he has prepared.

```

projectSGBD_create.sql projectSGBD_6-9.sql projectSGBD_10-14.sql projectSGBD_ex14.sql
SQL Worksheet History
projectSGBD
Worksheet Query Builder
END LOOP;
DBMS_OUTPUT.NEW_LINE;
END LOOP;

--daca numarul liniilor este 0, atunci inseamna ca bucatarul nu a gatit nimic inca
IF c_prod%ROWCOUNT = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Bucatarul dat nu a gatit niciun produs pana acum');
END IF;
CLOSE c_prod; --inchidere cursor

EXCEPTION
    WHEN NO_DATA_FOUND THEN --cazul in care nu exista niciun bucatar cu numele dat
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun bucatar cu numele dat');
    WHEN TOO_MANY_ROWS THEN --cazul in care exista mai multi bucatari cu numele dat
        RAISE_APPLICATION_ERROR(-20001, 'Există mai mulți bucătari cu numele dat');
END produse_gatite;
/

BEGIN
    produse_gatite('Avram', 'Alina');
    --produse_gatite('Stroilescu', 'Maria'); -- Bucatarul dat nu a gatit niciun produs pana acum
    --produse_gatite('Ana', 'Maria'); -- exceptie no_data_found (nu exista niciun bucatar cu acest nume)
    --produse_gatite('Brezan', 'Sabin-Alexandru'); -- exceptie too_many_rows (există doi bucătari cu acest nume)
END;
/
--pentru a introduce numele bucatarului de la tastatura
DECLARE

```

Output window:

```

BUCATARUL Avram Alina
Produsul: paste bolognese
-> spaghetti
-> sos de rosii
-> carne tocată

Produsul: pizza capriciosa
-> sos de rosii
-> parmezan
-> faina
-> masline

```

Script Result:

```

ORA-06512: at line 4

PL/SQL procedure successfully completed.

```

Exemplu apelare procedură pentru un bucătar care există, are un nume unic și nu a gătit produse până acum:

```

projectSGBD_create.sql projectSGBD_6-9.sql projectSGBD_10-14.sql projectSGBD_ex14.sql
SQL Worksheet History
projectSGBD
Worksheet Query Builder
END LOOP;
DBMS_OUTPUT.NEW_LINE;
END LOOP;

--daca numarul liniilor este 0, atunci inseamna ca bucatarul nu a gatit nimic inca
IF c_prod%ROWCOUNT = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Bucatarul dat nu a gatit niciun produs pana acum');
END IF;
CLOSE c_prod; --inchidere cursor

EXCEPTION
    WHEN NO_DATA_FOUND THEN --cazul in care nu exista niciun bucatar cu numele dat
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun bucatar cu numele dat');
    WHEN TOO_MANY_ROWS THEN --cazul in care exista mai multi bucatari cu numele dat
        RAISE_APPLICATION_ERROR(-20001, 'Există mai mulți bucătari cu numele dat');
END produse_gatite;
/

BEGIN
    --produse_gatite('Avram', 'Alina');
    produse_gatite('Stroilescu', 'Maria'); -- Bucatarul dat nu a gatit niciun produs pana acum
    --produse_gatite('Ana', 'Maria'); -- exceptie no_data_found (nu exista niciun bucatar cu acest nume)
    --produse_gatite('Brezan', 'Sabin-Alexandru'); -- exceptie too_many_rows (există doi bucătari cu acest nume)
END;
/
--pentru a introduce numele bucătarului de la tastatura
DECLARE
    /
Query Result Script Output
Task completed in 0.052 seconds
PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

```

Exemplu apelare procedură pentru un bucătar care nu există:

```

projectSGBD_create.sql projectSGBD_6-9.sql projectSGBD_10-14.sql projectSGBD_ex14.sql
SQL Worksheet History
projectSGBD
Worksheet Query Builder
--daca numarul liniilor este 0, atunci inseamna ca bucatarul nu a gatit nimic inca
IF c_prod%ROWCOUNT = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Bucatarul dat nu a gatit niciun produs pana acum');
END IF;
CLOSE c_prod; --inchidere cursor

EXCEPTION
    WHEN NO_DATA_FOUND THEN --cazul in care nu exista niciun bucatar cu numele dat
        RAISE_APPLICATION_ERROR(-20000, 'Nu există niciun bucătar cu numele dat');
    WHEN TOO_MANY_ROWS THEN --cazul in care exista mai multi bucatari cu numele dat
        RAISE_APPLICATION_ERROR(-20001, 'Există mai mulți bucătari cu numele dat');
END produse_gatite;
/

BEGIN
    --produse_gatite('Avram', 'Alina');
    --produse_gatite('Stroilescu', 'Maria'); -- Bucatarul dat nu a gatit niciun produs pana acum
    produse_gatite('Ana', 'Maria'); -- exceptie no_data_found (nu există niciun bucătar cu acest nume)
    --produse_gatite('Brezan', 'Sabin-Alexandru'); -- exceptie too_many_rows (există doi bucătari cu acest nume)
END;
/
--pentru a introduce numele bucătarului de la tastatura
Query Result Script Output
Task completed in 0.065 seconds
Error report -
ORA-20000: Nu există niciun bucătar cu numele dat
ORA-06512: at "PROIECTSGBD.PRODUSE_GATITE", line 56
ORA-06512: at line 4
20000. 00000 - "%s"
*Cause: The stored procedure 'raise_application_error'
was called which causes this error to be generated.
*Action: Correct the problem as described in the error message or contact
the application administrator or DBA for more information.

```

Exemplu apelare procedură cu un nume care corespunde pentru mai mulți bucătari:

```

--daca numarul liniilor este 0, atunci inseamna ca bucatarul nu a gatit nimic inca
IF c_prod%ROWCOUNT = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Bucatarul dat nu a gatit niciun produs pana acum');
END IF;
CLOSE c_prod; --inchidere cursor

EXCEPTION
    WHEN NO_DATA_FOUND THEN --cazul in care nu exista niciun bucatar cu numele dat
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun bucatar cu numele dat');
    WHEN TOO_MANY_ROWS THEN --cazul in care exista mai multi bucatari cu numele dat
        RAISE_APPLICATION_ERROR(-20001, 'Există mai mulți bucatari cu numele dat');
END produse_gatite;
/


BEGIN
    --produse_gatite('Avram', 'Alina');
    --produse_gatite('Stroilescu', 'Maria'); -- Bucatarul dat nu a gatit niciun produs pana acum
    --produse_gatite('Ana', 'Maria'); -- exceptie no_data_found (nu exista niciun bucatar cu acest nume)
    produse_gatite('Brezan', 'Sabin-Alexandru'); -- exceptie too_many_rows (există doi bucatari cu acest nume)
END;
/
--pentru a introduce numele bucatarului de la tastatura

```

Query Result | Script Output | Task completed in 0.083 seconds

```

--produse_gatite('Stroilescu', 'Maria'); -- Bucatarul dat nu a gatit niciun produs pana acum
--produse_gatite('Ana', 'Maria'); -- exceptie no_data_found (nu exista niciun bucatar cu acest nume)
produse_gatite('Brezan', 'Sabin-Alexandru'); -- exceptie too_many_rows (există doi bucatari cu acest nume)

END;
Error report -
ORA-20001: Există mai mulți bucatari cu numele dat
ORA-06512: at "PROIECTSGBD.PRODUSE_GATITE", line 58
ORA-06512: at line 5

```

8. Subprogram stocat de tip funcție care utilizează 3 tabele într-o comandă SQL

Definiți o funcție care primește ca parametru ziua unui eveniment și o categorie de produse și returnează numărul de produse comandate din acea categorie pentru evenimentul din ziua dată.

```

1  CREATE OR REPLACE FUNCTION nr_produse
2    (zi DATE,
3     categ categorii.nume_categorie%TYPE)
4  RETURN NUMBER IS
5    nr NUMBER := 0;
6    nr_aux NUMBER;
7    TYPE t_ind IS TABLE OF produse.id_produs%TYPE
8      INDEX BY PLS_INTEGER;
9    v_ind t_ind; --tablou indexat pentru a retine produsele ce fac parte din
10   -- categoria data
11   id_ev evenimente.id_eveniment%TYPE;
12   exc_categ EXCEPTION;
13   exc_produse EXCEPTION;
14
15  BEGIN
16    --verific daca exista categoria
17    SELECT COUNT(id_categorie)
18      INTO nr
19      FROM categorii
20      WHERE upper(nume_categorie) = upper(categ);
21
22    IF nr = 0 THEN --cazul in care categoria data nu exista
23      RAISE exc_categ;
24    END IF;
25
26    --verific daca exista produse in categoria data
27    SELECT COUNT(id_produs)
28      INTO nr
29      FROM produse p JOIN categorii c ON (p.id_categorie=c.id_categorie)
30      WHERE upper(nume_categorie) = upper(categ);
31
32    IF nr = 0 THEN --cazul in care nu exista produse in categoria data
33      RAISE exc_produse;
34    END IF;

```

```

34  --selectez id-urile produselor din categoria data
35  SELECT id_produs
36  BULK COLLECT INTO v_ind
37  FROM produse p JOIN categorii c ON (p.id_categorie=c.id_categorie)
38  WHERE upper(nume_categorie) = upper(categ);
39
40  --selectez id-ul evenimentului din ziua data
41  SELECT id_eveniment
42  INTO id_ev
43  FROM evenimente
44  WHERE data_eveniment = zi;
45
46  nr := 0;
47  FOR i IN v_ind.FIRST..v_ind.LAST LOOP
48    SELECT SUM(cantitate) --retin cantitatea vanduta pentru fiecare produs
        in parte
49    INTO nr_aux
50    FROM gateste_serveste g JOIN comenzi c ON (g.id_comanda = c.id_comanda)
51                  JOIN angajati a ON (g.id_angajat = a.id_angajat)
52    WHERE c.id_eveniment = id_ev AND tip_angajat = 'ospatar'
53      AND id_produs = v_ind(i);
54
55    nr := nr + NVL(nr_aux, 0); -- adaug cantitatea la suma totala
56  END LOOP;
57
58  RETURN nr; --returnez numarul de produse obtinut
59
60  EXCEPTION
61    WHEN NO_DATA_FOUND THEN --cazul in care nu exista un eveniment in ziua
       data
62      RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun eveniment in ziua
       data');
63    WHEN TOO_MANY_ROWS THEN --cazul in care exista mai multe evenimente in
       ziua data
64      RAISE_APPLICATION_ERROR(-20001, 'Există mai multe evenimente in ziua
       data');
65    WHEN exc_categ THEN --cazul in care categoria introdusa nu exista sau
       cazul in care inca nu sunt introduse produsele care fac parte din
       aceasta
66      RAISE_APPLICATION_ERROR(-20002, 'Categoria data nu exista');
67    WHEN exc_produse THEN --cazul in care inca nu sunt introduse produse
       care sa faca parte din categoria data
68      RAISE_APPLICATION_ERROR(-20003, 'Nu exista produse care sa faca parte
       din categoria data');
69  END nr_produse;
70 /
71

```

```

72|| DECLARE
73|   nr_prod NUMBER;
74| BEGIN
75|   nr_prod := nr_produse(TO_DATE('03-APR-21'), 'bauturi'); --23 produse = 20
76|     sticle apa + 3 sticle suc
77|   --nr_prod := nr_produse(TO_DATE('23-APR-21'), 'salate'); --exceptie: Nu
78|     exista niciun eveniment in ziua data
79|   --nr_prod := nr_produse(TO_DATE('28-FEB-21'), 'paste'); -- exceptie: Exista
80|     mai multe evenimente in ziua data
81|   --nr_prod := nr_produse(TO_DATE('13-JUL-21'), 'aaa'); --exceptie: Categoria
82|     data nu exista
83|   --nr_prod := nr_produse(TO_DATE('13-JUL-21'), 'garnituri');-- exceptie: Nu
84|     exista produse care sa faca parte din categoria data
85| IF nr_prod = 0 THEN
86|   DBMS_OUTPUT.PUT_LINE('Nu au fost comandate produse din aceasta
87|                         categorie');
88| ELSE
89|   DBMS_OUTPUT.PUT_LINE('Au fost comandate ' || nr_prod || ' produse din
90|                         aceasta categorie');
91| END IF;
92| END;
93| /
94| --pentru a introduce data evenimentului si categoria de la tastatura
95| DECLARE
96|   nr_prod NUMBER;
97|   data_ev DATE := TO_DATE('&p_data');
98|   categ categorii.nume_categorie%TYPE := '&p_categoria';
99| BEGIN
100|   nr_prod := nr_produse(data_ev, categ);
101|   IF nr_prod = 0 THEN
102|     DBMS_OUTPUT.PUT_LINE('Nu au fost comandate produse din categoria ' ||
103|                           categ);
104|   ELSE
105|     DBMS_OUTPUT.PUT_LINE('Au fost comandate ' || nr_prod || ' produse din
106|                           categoria ' || categ);
107|   END IF;
108| END;
109| /

```

Exemplu apelare funcție:

```

EXCEPTION
    WHEN NO_DATA_FOUND THEN --cazul in care nu exista un eveniment in ziua data
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun eveniment in ziua data');
    WHEN TOO_MANY_ROWS THEN --cazul in care exista mai multe evenimente in ziua data
        RAISE_APPLICATION_ERROR(-20001, 'Există mai multe evenimente in ziua data');
    WHEN exc_categ THEN --cazul in care categoria introdusa nu exista sau cauzul in care inca nu sunt produse
        RAISE_APPLICATION_ERROR(-20002, 'Categoria data nu există');
    WHEN exc_produse THEN --cazul in care inca nu sunt introduce produse care sa faca parte din categoria data
        RAISE_APPLICATION_ERROR(-20003, 'Nu există produse care sa faca parte din categoria data');
END nr_produse;
/

DECLARE
    nr_prod NUMBER;
BEGIN
    nr_prod := nr_produse(TO_DATE('03-APR-21'), 'bauturi'); --23 produse = 20 sticle apa + 3 sticle suc
    --nr_prod := nr_produse(TO_DATE('23-APR-21'), 'salate'); --exceptie: Nu exista niciun eveniment in ziua data
    --nr_prod := nr_produse(TO_DATE('28-FEB-21'), 'paste'); -- exceptie: Există mai multe evenimente in ziua data
    --nr_prod := nr_produse(TO_DATE('13-JUL-21'), 'aaa'); --exceptie: Categoria data nu există
    --nr_prod := nr_produse(TO_DATE('13-JUL-21'), 'garnituri');-- exceptie: Nu există produse care sa faca parte din categoria data
    IF nr_prod = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Nu au fost comandate produse din aceasta categorie');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Au fost comandate ' || nr_prod || ' produse din aceasta categorie');
    END IF;
END;
/

```

PL/SQL procedure successfully completed.

Exemplu apelare funcție cu o dată la care nu este programat un eveniment:

```

RAISE_APPLICATION_ERROR(20001, 'Există mai multe evenimente in ziua data');
WHEN exc_categ THEN --cazul in care categoria introdusa nu exista sau cauzul in care inca nu sunt introduce produsele care fac parte din categoria data
    RAISE_APPLICATION_ERROR(-20002, 'Categoria data nu există');
WHEN exc_produse THEN --cazul in care inca nu sunt introduce produse care sa faca parte din categoria data
    RAISE_APPLICATION_ERROR(-20003, 'Nu există produse care sa faca parte din categoria data');
END nr_produse;
/

DECLARE
    nr_prod NUMBER;
BEGIN
    --nr_prod := nr_produse(TO_DATE('03-APR-21'), 'bauturi'); --23 produse = 20 sticle apa + 3 sticle suc
    nr_prod := nr_produse(TO_DATE('23-APR-21'), 'salate'); --exceptie: Nu exista niciun eveniment in ziua data
    --nr_prod := nr_produse(TO_DATE('28-FEB-21'), 'paste'); -- exceptie: Există mai multe evenimente in ziua data
    --nr_prod := nr_produse(TO_DATE('13-JUL-21'), 'aaa'); --exceptie: Categoria data nu există
    --nr_prod := nr_produse(TO_DATE('13-JUL-21'), 'garnituri');-- exceptie: Nu există produse care sa faca parte din categoria data
    IF nr_prod = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Nu au fost comandate produse din aceasta categorie');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Au fost comandate ' || nr_prod || ' produse din aceasta categorie');
    END IF;
END;
/

```

Task completed in 0.364 seconds

END;

Error report -

ORA-20000: Nu exista niciun eveniment in ziua data
ORA-06512: at "PROJECTSGBD.NR_PRODUSE", line 62
ORA-06512: at line 5
20000. 00000 - "%s"
*Cause: The stored procedure 'raise_application_error'
was called which causes this error to be generated.
*Action: Correct the problem as described in the error message or contact
the application administrator or DBA for more information.

Exemplu apelare funcție cu o dată la care sunt programate mai multe evenimente:

```

RAISE_APPLICATION_ERROR(-20001, 'Există mai multe evenimente în ziua data');
WHEN exc_categ THEN --cazul în care categoria introdusa nu există sau cazul în care încă nu sunt introduse produsele din categoria
    RAISE_APPLICATION_ERROR(-20002, 'Categoria data nu există');
WHEN exc_produc THEN --cazul în care încă nu sunt introduse produse care să facă parte din categoria data
    RAISE_APPLICATION_ERROR(-20003, 'Nu există produse care să facă parte din categoria data');

END nr_produc;
/

DECLARE
    nr_prod NUMBER;
BEGIN
    --nr_prod := nr_produc(TO_DATE('03-APR-21'), 'bauturi'); --23 produse = 20 sticle apa + 3 sticle suc
    --nr_prod := nr_produc(TO_DATE('23-APR-21'), 'salate'); --excepție: Nu există niciun eveniment în ziua data
    nr_prod := nr_produc(TO_DATE('28-FEB-21'), 'paste'); -- excepție: Există mai multe evenimente în ziua data
    --nr_prod := nr_produc(TO_DATE('13-JUL-21'), 'aaa'); --excepție: Categoria data nu există
    --nr_prod := nr_produc(TO_DATE('13-JUL-21'), 'garnituri');-- excepție: Nu există produse care să facă parte din categoria
    IF nr_prod = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Nu au fost comandate produse din aceasta categorie');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Au fost comandate ' || nr_prod || ' produse din aceasta categorie');
    END IF;
END;
/

```

Script Output x

Task completed in 0.3 seconds

```

ELSE
    DBMS_OUTPUT.PUT_LINE('Au fost comandate ' || nr_prod || ' produse din aceasta categorie');
END IF;
END;
Error report -
ORA-20001: Există mai multe evenimente în ziua data
ORA-06512: at "PROIECTSGBD.NR_PRODUSE", line 64
ORA-06512: at line 6

```

Exemplu apelare funcție cu o categorie care nu există:

```

RAISE_APPLICATION_ERROR(-20001, 'Există mai multe evenimente în ziua data');
WHEN exc_categ THEN --cazul în care categoria introdusa nu există sau cazul în care încă nu sunt introduse produsele din categoria
    RAISE_APPLICATION_ERROR(-20002, 'Categoria data nu există');
WHEN exc_produc THEN --cazul în care încă nu sunt introduse produse care să facă parte din categoria data
    RAISE_APPLICATION_ERROR(-20003, 'Nu există produse care să facă parte din categoria data');

END nr_produc;
/

DECLARE
    nr_prod NUMBER;
BEGIN
    --nr_prod := nr_produc(TO_DATE('03-APR-21'), 'bauturi'); --23 produse = 20 sticle apa + 3 sticle suc
    --nr_prod := nr_produc(TO_DATE('23-APR-21'), 'salate'); --excepție: Nu există niciun eveniment în ziua data
    --nr_prod := nr_produc(TO_DATE('28-FEB-21'), 'paste'); -- excepție: Există mai multe evenimente în ziua data
    nr_prod := nr_produc(TO_DATE('13-JUL-21'), 'aaa'); --excepție: Categoria data nu există
    --nr_prod := nr_produc(TO_DATE('13-JUL-21'), 'garnituri');-- excepție: Nu există produse care să facă parte din categoria
    IF nr_prod = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Nu au fost comandate produse din aceasta categorie');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Au fost comandate ' || nr_prod || ' produse din aceasta categorie');
    END IF;
END;
/

```

Script Output x

Task completed in 0.094 seconds

```

ELSE
    DBMS_OUTPUT.PUT_LINE('Au fost comandate ' || nr_prod || ' produse din aceasta categorie');
END IF;
END;
Error report -
ORA-20002: Categoria data nu există
ORA-06512: at "PROIECTSGBD.NR_PRODUSE", line 66
ORA-06512: at line 7

```

Exemplu apelare funcție cu o categorie pentru care încă nu este introdus niciun produs:

```

projectSGBD_create.sql * projectSGBD_6-9.sql * projectSGBD_10-14.sql *
SQL Worksheet History
Query Builder
projectSGBD

RAISE_APPLICATION_ERROR(-20001, 'Există mai multe evenimente în ziua data');
WHEN exc_Categ THEN --cauză în care categoria introdusa nu există sau cauză în care încă nu sunt introduse produsele din categoria data
    RAISE_APPLICATION_ERROR(-20002, 'Categoria data nu există');
WHEN exc_Produse THEN --cauză în care încă nu sunt introduse produse care să facă parte din categoria data
    RAISE_APPLICATION_ERROR(-20003, 'Nu există produse care să facă parte din categoria data');

END nr_Produse;
/

DECLARE
    nr_Prod NUMBER;
BEGIN
    --nr_Prod := nr_Produse(TO_DATE('03-APR-21'), 'bauturi'); --23 produse = 20 sticle apa + 3 sticle suc
    --nr_Prod := nr_Produse(TO_DATE('23-APR-21'), 'salate'); --excepție: Nu există niciun eveniment în ziua data
    --nr_Prod := nr_Produse(TO_DATE('28-FEB-21'), 'paste'); -- excepție: Există mai multe evenimente în ziua data
    --nr_Prod := nr_Produse(TO_DATE('13-JUL-21'), 'aaa'); --excepție: Categoria data nu există
    nr_Prod := nr_Produse(TO_DATE('13-JUL-21'), 'garnituri');-- excepție: Nu există produse care să facă parte din categoria data
    IF nr_Prod = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Nu au fost comandate produse din această categorie');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Au fost comandate ' || nr_Prod || ' produse din această categorie');
    END IF;
END;
/

```

Script Output x

Task completed in 0.084 seconds

```

Error report -
ORA-20003: Nu există produse care să facă parte din categoria data
ORA-06512: at "PROIECTSGBD.NR_PRODUSE", line 68
ORA-06512: at line 8

```

9. Subprogram stocat de tip procedură care utilizează 5 tabele într-o comandă SQL

Să se scrie o procedură care pentru numele unui client dat să afișeze pentru fiecare eveniment toți furnizorii care au furnizat ingrediente ce au fost conținute de produsele din comenzi date pentru evenimentul respectiv.

În rezolvarea exercițiului am folosit un cursor imbricat. Cursorul părinte încarcă date despre evenimentele clientului și un cursor generat de o subcerere pentru a încărca și procesa furnizorii de ingrediente folosite pentru fiecare eveniment. Subcererea utilizează 5 dintre tabelele din baza de date.

```

1  CREATE OR REPLACE PROCEDURE furnizori_evenimente
2      (v_nume clienti.nume%TYPE,
3       v_prenume clienti.prenume%TYPE)
4  IS
5      --cursor imbricat
6      CURSOR ev(id_c clienti.id_client%TYPE) IS
7          SELECT data_eveniment, denumire_tip, --selectez date despre
8                  eveniment
9              CURSOR(SELECT DISTINCT f.denumire --selectez
10                     furnizorii ceruti in functie de eveniment
11                     FROM furnizori f JOIN ingrediente i ON
12                         (f.id_furnizor=i.id_furnizor)
13                     JOIN contine c ON
14                         (i.id_ingredient=c.id_ingredient)
15                     JOIN produse p ON
16                         (p.id_produs=c.id_produs)
17                     JOIN gateste_serveste g ON
18                         (p.id_produs=g.id_produs)
19                     WHERE g.id_comanda = c.id_comanda
20             )
21     FROM evenimente e JOIN tipuri_evenimente t ON
22         (e.id_tip_eveniment = t.id_tip_eveniment)
23         JOIN comenzi c ON (e.id_eveniment =
24             c.id_eveniment)
25         JOIN clienti cli ON (cli.id_client = c.id_client)
26         WHERE cli.id_client = id_c;
27
28  c_furnizori SYS_REFCURSOR;

```

```

21    v_id clienti.id_client%TYPE;
22    data_ev evenimente.data_eveniment%TYPE;
23    nume_tip tipuri_evenimente.denumire_tip%TYPE;
24    TYPE tablou_indexat IS TABLE OF furnizori.denumire%TYPE INDEX BY PLS_INTEGER;
25    t_furnizori tablou_indexat; --tablou indexat pt a retine denumirea
26        furnizorilor
27    exc EXCEPTION;
28 BEGIN
29     SELECT id_client --selectez id-ul clientului introdus
30       INTO v_id
31     FROM clienti
32     WHERE upper(nume)=upper(v_nume) AND upper(prenume)=upper(v_prenume);
33
34     OPEN ev(v_id); --deschid cursorul
35
36 LOOP
37     FETCH ev INTO data_ev, nume_tip, c_furnizori;
38     EXIT WHEN ev%NOTFOUND;
39
40     DBMS_OUTPUT.PUT_LINE('Eveniment: ' || nume_tip || ' din data ' ||
41                           TO_CHAR(data_ev));
42     FETCH c_furnizori BULK COLLECT INTO t_furnizori;
43
44     IF t_furnizori.COUNT = 0 THEN
45         DBMS_OUTPUT.PUT_LINE('Pentru acest eveniment nu au fost date
46                               comenzi');
47     ELSE
48         FOR i IN t_furnizori.FIRST..t_furnizori.LAST LOOP
49             DBMS_OUTPUT.PUT_LINE(i|| '. '|| t_furnizori(i)); --afisez
50                 furnizorii
51         END LOOP;
52     END IF;
53
54     DBMS_OUTPUT.NEW_LINE;
55 END LOOP;
56
57 IF ev%ROWCOUNT = 0 THEN --cazul in care clientul nu a dat nicio comanda
58     pentru un eveniment
59     RAISE exc;
60 END IF;
61
62 CLOSE ev;
63 EXCEPTION
64     WHEN NO_DATA_FOUND THEN -- cazul in care nu exista niciun client cu
65         numele dat
66         RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun client cu numele
67             dat');

```

```

61    WHEN TOO_MANY_ROWS THEN -- cazul in care exista mai multi clienti cu
       numele dat
62        RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi clienti cu numele
           dat');
63    WHEN exc THEN -- cazul in care clientul nu a dat nicio comanda pentru un
       eveniment
64        RAISE_APPLICATION_ERROR(-20002, 'Clientul dat nu a dat nicio comanda
           pentru un eveniment');
65 END furnizori_evenimente;
66 /
67
68 BEGIN
69     furnizori_evenimente('Popescu', 'Maria');
70     --furnizori_evenimente('Ana', 'Ana'); --exceptie: Nu exista niciun client cu
       numele dat
71     --furnizori_evenimente('Marin', 'Liviu'); --exceptie: Exista mai multi
       clienti cu numele dat
72     --furnizori_evenimente('Gavrilă', 'Cristina'); --exceptie: Clientul dat nu a
       dat nicio comanda pentru un eveniment
73 END;
74 /
75 --pentru a introduce numele clientului de la tastatura
76 DECLARE
77     nume clienti.nume%TYPE := '&p_nume';
78     prenume clienti.prenume%TYPE := '&p_prenume';
79 BEGIN
80     furnizori_evenimente(nume, prenume);
81 END;
82 /

```

Exemplu apelare procedură:

```

DBMS_OUTPUT.NEW_LINE;
END LOOP;

IF ev%ROWCOUNT = 0 THEN --cazul in care clientul nu a dat nicio comanda pentru un eveniment
    RAISE exc;
END IF;

CLOSE ev;
EXCEPTION
    WHEN NO_DATA_FOUND THEN -- cazul in care nu exista niciun client cu numele dat
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun client cu numele dat');
    WHEN TOO_MANY_ROWS THEN -- cazul in care exista mai multi clienti cu numele dat
        RAISE_APPLICATION_ERROR(-20001, 'Există mai mulți clienti cu numele dat');
    WHEN exc THEN -- cazul in care clientul nu a dat nicio comanda pentru un eveniment
        RAISE_APPLICATION_ERROR(-20002, 'Clientul dat nu a dat nicio comanda pentru un eveniment');
END furnizori_evenimente;
/
BEGIN
    furnizori_evenimente('Popescu', 'Maria');
    --furnizori_evenimente('Ana', 'Ana'); --exceptie: Nu exista niciun client cu numele dat
    --furnizori_evenimente('Marin', 'Liviu'); --exceptie: Există mai mulți clienti cu numele dat
    --furnizori_evenimente('Gavrilă', 'Cristina'); --exceptie: Clientul dat nu a dat nicio comanda per
END;
/

```

Procedure FURNIZORI_EVENIMENTE compiled

PL/SQL procedure successfully completed.

Exemplu apelare procedură cu numele unui client care nu există:

```

DBMS_OUTPUT.NEW_LINE;
END LOOP;

IF ev%ROWCOUNT = 0 THEN --cazul in care clientul nu a dat nicio comanda pentru un eveniment
    RAISE exc;
END IF;

CLOSE ev;
EXCEPTION
    WHEN NO_DATA_FOUND THEN -- cazul in care nu exista niciun client cu numele dat
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun client cu numele dat');
    WHEN TOO_MANY_ROWS THEN -- cazul in care exista mai multi clienti cu numele dat
        RAISE_APPLICATION_ERROR(-20001, 'Există mai mulți clienti cu numele dat');
    WHEN exc THEN -- cazul in care clientul nu a dat nicio comanda pentru un eveniment
        RAISE_APPLICATION_ERROR(-20002, 'Clientul dat nu a dat nicio comanda pentru un eveniment');
END furnizori_evenimente;
/
BEGIN
    --furnizori_evenimente('Popescu', 'Maria');
    furnizori_evenimente('Ana', 'Ana'); --exceptie: Nu exista niciun client cu numele dat
    --furnizori_evenimente('Marin', 'Liviu'); --exceptie: Există mai mulți clienti cu numele dat
    --furnizori_evenimente('Gavrilă', 'Cristina'); --exceptie: Clientul dat nu a dat nicio comanda pentru un eveniment
END;
/

```

Error report -

ORA-20000: Nu exista niciun client cu numele dat
ORA-06512: at "PROJECTSGBD.FURNIZORI_EVENIMENTE", line 60
ORA-06512: at line 3
20000. 00000 - "%s"
*Cause: The stored procedure 'raise_application_error'
was called which causes this error to be generated.
*Action: Correct the problem as described in the error message or contact
the application administrator or DBA for more information.

Exemplu apelare procedură cu un nume care corespunde pentru mai mulți clienți:

```

projectSGBD_create.sql projectSGBD_6-9.sql projectSGBD_10-14.sql
SQL Worksheet History
Worksheet Query Builder
END IF;

DBMS_OUTPUT.NEW_LINE;
END LOOP;

IF ev%ROWCOUNT = 0 THEN --cazul in care clientul nu a dat nicio comanda pentru un eveniment
    RAISE exc;
END IF;

CLOSE ev;
EXCEPTION
    WHEN NO_DATA_FOUND THEN -- cazul in care nu exista niciun client cu numele dat
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun client cu numele dat');
    WHEN TOO_MANY_ROWS THEN -- cazul in care exista mai multi clienti cu numele dat
        RAISE_APPLICATION_ERROR(-20001, 'Există mai multi clienti cu numele dat');
    WHEN exc THEN -- cazul in care clientul nu a dat nicio comanda pentru un eveniment
        RAISE_APPLICATION_ERROR(-20002, 'Clientul dat nu a dat nicio comanda pentru un eveniment');
END furnizori_evenimente;
/
BEGIN
    --furnizori_evenimente('Popescu', 'Maria');
    --furnizori_evenimente('Ana', 'Ana'); --exceptie: Nu exista niciun client cu numele dat
    furnizori_evenimente('Marin', 'Liviu'); --exceptie: Există mai multi clienti cu numele dat
    --furnizori_evenimente('Gavrilă', 'Cristina'); --exceptie: Clientul dat nu a dat nicio comanda pentru un eveniment
END;

```

Script Output | Query Result | Task completed in 0.075 seconds

```

--furnizori_evenimente('Gavrilă', 'Cristina'); --exceptie: Clientul dat nu a dat nicio comanda pentru un eveniment
END;
Error report -
ORA-20001: Există mai multi clienti cu numele dat
ORA-06512: at "PROIECTSGBD.FURNIZORI_EVENIMENTE", line 62
ORA-06512: at line 4

```

Exemplu apelare procedură cu numele unui client care nu a dat nicio comandă pentru un eveniment:

```

projectSGBD_create.sql projectSGBD_6-9.sql projectSGBD_10-14.sql
SQL Worksheet History
Worksheet Query Builder
END IF;

DBMS_OUTPUT.NEW_LINE;
END LOOP;

IF ev%ROWCOUNT = 0 THEN --cazul in care clientul nu a dat nicio comanda pentru un eveniment
    RAISE exc;
END IF;

CLOSE ev;
EXCEPTION
    WHEN NO_DATA_FOUND THEN -- cazul in care nu exista niciun client cu numele dat
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun client cu numele dat');
    WHEN TOO_MANY_ROWS THEN -- cazul in care exista mai multi clienti cu numele dat
        RAISE_APPLICATION_ERROR(-20001, 'Există mai multi clienti cu numele dat');
    WHEN exc THEN -- cazul in care clientul nu a dat nicio comanda pentru un eveniment
        RAISE_APPLICATION_ERROR(-20002, 'Clientul dat nu a dat nicio comanda pentru un eveniment');
END furnizori_evenimente;
/
BEGIN
    --furnizori_evenimente('Popescu', 'Maria');
    --furnizori_evenimente('Ana', 'Ana'); --exceptie: Nu exista niciun client cu numele dat
    --furnizori_evenimente('Marin', 'Liviu'); --exceptie: Există mai multi clienti cu numele dat
    furnizori_evenimente('Gavrilă', 'Cristina'); --exceptie: Clientul dat nu a dat nicio comanda pentru un eveniment
END;

```

Script Output | Query Result | Task completed in 0.091 seconds

```

furnizori_evenimente('Gavrilă', 'Cristina'); --exceptie: Clientul dat nu a dat nicio comanda pentru un eveniment
END;
Error report -
ORA-20002: Clientul dat nu a dat nicio comanda pentru un eveniment
ORA-06512: at "PROIECTSGBD.FURNIZORI_EVENIMENTE", line 64
ORA-06512: at line 5

```

10. Trigger de tip LMD la nivel de comandă

Creați un trigger care nu permite adăugarea, ștergerea și modificarea comenziilor în afara programului restaurantului. În timpul săptămânii restaurantul este deschis între 10:00-22:00, iar în weekend între 9:00-23:30.

```

1  CREATE OR REPLACE TRIGGER program_restaurant
2    BEFORE INSERT OR UPDATE OR DELETE ON comenzi
3    BEGIN
4      IF TO_CHAR(SYSDATE, 'DAY') = 'SATURDAY' OR TO_CHAR(SYSDATE, 'DAY') =
5        'SUNDAY' THEN -- in weekend, programul este 9 - 23:30
6          IF TO_NUMBER(TO_CHAR(SYSDATE, 'HH24')) < 9 THEN
7              RAISE_APPLICATION_ERROR(-20110, 'Restaurantul se deschide la 9:00 in
8                weekend!');
9          END IF;
10         IF TO_NUMBER(TO_CHAR(SYSDATE, 'HH24')) = 23 AND
11             TO_NUMBER(TO_CHAR(SYSDATE, 'MI')) > 30 THEN
12                 RAISE_APPLICATION_ERROR(-20111, 'Restaurantul este inchis dupa ora
13                   23:30 in weekend!');
14             END IF;
15         ELSE -- in timpul saptamanii, programul este 10 - 22
16             IF TO_NUMBER(TO_CHAR(SYSDATE, 'HH24')) < 10 THEN
17                 RAISE_APPLICATION_ERROR(-20112, 'Restaurantul se deschide la 9:00 in
18                   timpul saptamanii!');
19             END IF;
20             IF TO_NUMBER(TO_CHAR(SYSDATE, 'HH24')) >= 22 THEN
21                 RAISE_APPLICATION_ERROR(-20113, 'Restaurantul este inchis dupa ora
22                   22:00 in timpul saptamanii!');
23             END IF;
24         END IF;
25     END;
26   /
27   BEGIN
28     DBMS_OUTPUT.PUT_LINE('Zi: ' || TO_CHAR(SYSDATE, 'DAY'));
29     DBMS_OUTPUT.PUT_LINE('Ora: ' || TO_CHAR(SYSDATE, 'HH24:MI'));
30     INSERT INTO comenzi
31       VALUES(SEQ_COMENZI.NEXTVAL, to_date('15-12-2021', 'dd-mm-yyyy'), 610, 7,
32             null);
33   END;
34   /

```

Exemplu inserare în timpul programului:

The screenshot shows the Oracle SQL Developer interface. In the top tab bar, there are three tabs: projectSGBD_create.sql, projectSGBD_6-9.sql, and projectSGBD_10-14.sql. The main workspace contains a PL/SQL code editor with the following trigger definition:

```

CREATE OR REPLACE TRIGGER program_restaurant
  BEFORE INSERT OR UPDATE OR DELETE ON comenzi
AS
BEGIN
  IF TO_CHAR(SYSDATE, 'DAY') = 'SATURDAY' OR TO_CHAR(SYSDATE, 'DAY') = 'SUNDAY' THEN
    -- in weekend, programul este 9 - 23:30
    IF TO_NUMBER(TO_CHAR(SYSDATE, 'HH24')) < 9 THEN
      RAISE_APPLICATION_ERROR(-20110, 'Restaurantul se deschide la 9:00 in weekend!');
    END IF;
    IF TO_NUMBER(TO_CHAR(SYSDATE, 'HH24')) = 23 AND TO_NUMBER(TO_CHAR(SYSDATE, 'MI')) > 30 THEN
      RAISE_APPLICATION_ERROR(-20111, 'Restaurantul este inchis dupa ora 23:30 in weekend!');
    END IF;
  ELSE
    -- in timpul saptamanii, programul este 10 - 22
    IF TO_NUMBER(TO_CHAR(SYSDATE, 'HH24')) < 10 THEN
      RAISE_APPLICATION_ERROR(-20112, 'Restaurantul se deschide la 9:00 in timpul saptamanii!');
    END IF;
    IF TO_NUMBER(TO_CHAR(SYSDATE, 'HH24')) >= 22 THEN
      RAISE_APPLICATION_ERROR(-20113, 'Restaurantul este inchis dupa ora 22:00 in timpul saptamanii!');
    END IF;
  END IF;
END;
/
BEGIN
  DBMS_OUTPUT.PUT_LINE('Zi: ' || TO_CHAR(SYSDATE, 'DAY'));
  DBMS_OUTPUT.PUT_LINE('Ora: ' || TO_CHAR(SYSDATE, 'HH24:MI'));
  INSERT INTO comenzi
    VALUES(SEQ_COMENZI.NEXTVAL, to_date('15-12-2021', 'dd-mm-yyyy'), 610, 7, null);
END;

```

The cursor is positioned over the last line of the trigger body. The right pane shows the database connection details: projectSGBD, Zi: WEDNESDAY, Ora: 20:56. Below the workspace, the status bar indicates "Task completed in 0.046 seconds".

Inserarea este efectuată cu succes:

The screenshot shows the Oracle SQL Developer interface with a query result window. The query executed is:

```
SELECT * FROM comenzi;
```

The results are displayed in a table:

ID_COMANDA	DATA_COMANDA	ID_MOD_PLATA	ID_CLIENT	ID_EVENIMENT
1	10 09-MAR-21	600	2	113
2	20 15-OCT-20	610	5	(null)
3	30 03-MAY-20	620	1	(null)
4	40 05-JUL-21	600	3	110
5	50 18-OCT-19	630	7	(null)
6	60 23-DEC-20	610	4	(null)
7	70 16-FEB-21	620	2	114
8	80 30-APR-21	600	6	111
9	90 02-AUG-20	630	1	112
10	100 05-OCT-21	600	1	116
11	120 15-DEC-21	610	7	(null)

Exemplu inserare în afara programului, inserarea nu este efectuată pentru că este declanșat trigger-ul:

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there are three tabs: 'projectSGBD_create.sql', 'projectSGBD_6-9.sql', and 'projectSGBD_10-14.sql'. The 'projectSGBD' connection is selected. In the central 'Worksheet' tab, a PL/SQL block is displayed:

```

    RAISE_APPLICATION_ERROR(-20110, 'Restaurantul se deschide la 9:00 in weekend!');
END IF;
IF TO_NUMBER(TO_CHAR(SYSDATE, 'HH24')) = 23 AND TO_NUMBER(TO_CHAR(SYSDATE, 'MI')) > 30 THEN
    RAISE_APPLICATION_ERROR(-20111, 'Restaurantul este inchis dupa ora 23:30 in weekend!');
END IF;
ELSE
    -- in timpul saptamanii, programul este 10 - 22
    IF TO_NUMBER(TO_CHAR(SYSDATE, 'HH24')) < 10 THEN
        RAISE_APPLICATION_ERROR(-20112, 'Restaurantul se deschide la 9:00 in timpul saptamanii!');
    END IF;
    IF TO_NUMBER(TO_CHAR(SYSDATE, 'HH24')) >= 22 THEN
        RAISE_APPLICATION_ERROR(-20113, 'Restaurantul este inchis dupa ora 22:00 in timpul saptamanii!');
    END IF;
END IF;
END;
/
BEGIN
    DBMS_OUTPUT.PUT_LINE('Zi: ' || TO_CHAR(SYSDATE, 'DAY'));
    DBMS_OUTPUT.PUT_LINE('Ora: ' || TO_CHAR(SYSDATE, 'HH24:MI'));
    INSERT INTO comenzi
        VALUES(SEQ_COMENZI.NEXTVAL, to_date('15-12-2021','dd-mm-yyyy'), 610, 7, null);
END;
/

```

In the bottom-left pane, the 'Script Output' tab shows the executed SQL statement and its result:

```

INSERT INTO comenzi
VALUES(SEQ_COMENZI.NEXTVAL, to_date('15-12-2021','dd-mm-yyyy'), 610, 7, null);
END;

Error report -
ORA-20113: Restaurantul este inchis dupa ora 22:00 in timpul saptamanii!
ORA-06512: at "PROJECTSGBD.PROGRAM_RESTAURANT", line 16
ORA-04080: error during execution of trigger 'PROJECTSGBD.PROGRAM_RESTAURANT'
ORA-06512: at line 4

```

The 'Dbms Output' tab in the top-right shows the current date and time:

Zi: WEDNESDAY
Ora: 22:04

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there are three tabs: 'projectSGBD_create.sql', 'projectSGBD_6-9.sql', and 'projectSGBD_10-14.sql'. The 'projectSGBD' connection is selected. In the central 'Worksheet' tab, a simple query is run:

```

SELECT * FROM comenzi;

```

In the bottom-left pane, the 'Query Result' tab shows the results of the query:

ID_COMANDA	DATA_COMANDA	ID_MOD_PLATA	ID_CLIENT	ID_EVENIMENT
1	10 09-MAR-21	600	2	113
2	2015-OCT-20	610	5	(null)
3	30 03-MAY-20	620	1	(null)
4	40 05-JUL-21	600	3	110
5	50 18-OCT-19	630	7	(null)
6	60 23-DEC-20	610	4	(null)
7	70 16-FEB-21	620	2	114
8	80 30-APR-21	600	6	111
9	90 02-AUG-20	630	1	112
10	100 05-OCT-21	600	1	116

11. Trigger de tip LMD la nivel de linie

Să se scrie un trigger care atunci când se încearcă servirea unor produse să verifice ca acestea să fi fost preparate înainte de un bucătar. Excepție fac produsele din categoria băuturi care pot fi servite direct, fără a fi pregătite de un bucătar. La ștergerea unei înregistrări să se afișeze detalii despre cantitatea ștearsă.

```

1  CREATE OR REPLACE TRIGGER prod_gatit
2    BEFORE INSERT OR DELETE
3    ON gateste_serveste
4    FOR EACH ROW
5    DECLARE
6      tip_angajati.tip_angajat%TYPE;
7      bauturi_id categorii.id_categorie%TYPE;
8      categ_id categorii.id_categorie%TYPE;
9      nr_prod_gatite NUMBER;
10     nr_prod_servite NUMBER;
11     can_stearsa gateste_serveste.cantitate%TYPE;
12
13    BEGIN
14      IF INSERTING THEN
15        --selectez tipul angajatului
16        SELECT tip_angajat
17          INTO tip
18        FROM angajati
19        WHERE id_angajat = :NEW.id_angajat;
20
21      IF tip = 'ospatar' THEN
22        --verificam din ce categorie face parte produsul
23        --numai bauturile nu trebuie gătite, deci verificam ca produsul sa nu
24        --faca parte din aceasta categorie
25        SELECT id_categorie --selectez id-ul categoriei 'bauturi'
26          INTO bauturi_id
27        FROM categorii
28        WHERE nume_categorie = 'bauturi';
29
30      SELECT id_categorie --selectez id-ul categoriei din care face parte
31        produsul
32          INTO categ_id
33        FROM produse
34        WHERE id_produs = :NEW.id_produs;

```

```

32
33     IF categ_id != bauturi_id THEN --produsul trebuie gatit
34         SELECT NVL(SUM(cantitate),0) --retin cantitatea de produse gatite
35             pentru aceasta comanda
36             INTO nr_prod_gatite
37             FROM gateste_serveste g JOIN angajati a ON (g.id_angajat =
38                 a.id_angajat)
39             WHERE id_produs = :NEW.id_produs AND id_comanda = :NEW.id_comanda
40                 AND tip_angajat = 'bucatar';
41
42             DBMS_OUTPUT.PUT_LINE('Nr produse gatite:' || nr_prod_gatite);
43
44             --calculam si numarul de produse deja servite
45             SELECT NVL(SUM(cantitate),0)
46                 INTO nr_prod_servite
47                 FROM gateste_serveste g JOIN angajati a ON (g.id_angajat =
48                     a.id_angajat)
49                 WHERE id_produs = :NEW.id_produs AND id_comanda = :NEW.id_comanda
50                     AND tip_angajat = 'ospatar';
51
52             DBMS_OUTPUT.PUT_LINE('Nr produse servite:' || nr_prod_servite);
53             DBMS_OUTPUT.PUT_LINE('Cantitatea care ar putea fi servita:' || 
54                 nr_prod_gatite - nr_prod_servite);
55
56             IF nr_prod_gatite - nr_prod_servite < :NEW.cantitate THEN
57                 RAISE_APPLICATION_ERROR(-20000, 'Cantitatea mentionata nu
58                     poate fi servita');
59             END IF;
60             END IF;
61             END IF;
62 ELSE
63     --incercam sa selectam cantitatea din tabelul in care se produce
64     --modificarea => tabel mutating
65     SELECT cantitate
66         INTO can_stearsa
67         FROM gateste_serveste
68         WHERE id_angajat = :OLD.id_angajat AND id_comanda = :OLD.id_comanda AND
69             id_produs = :OLD.id_produs;
70
71     --rezolvare fara a aparea eroarea 'table is mutating'
72     --can_stearsa := :OLD.cantitate;
73
74     DBMS_OUTPUT.PUT_LINE('Cantitatea ' || can_stearsa || ' a fost stearsa cu
75         succes');
76     END IF;
77 END;
78 /

```

```

71  ALTER TRIGGER prod_gatit DISABLE;
72  ALTER TRIGGER prod_gatit ENABLE;
73
74  --incercam servirea unor produse care nu sunt gatite => declansam trigger-ul
75  INSERT INTO gateste_serveste
76  VALUES(201, 30, 51, 5);
77
78  --cazul in care produsele sunt gatite inainte, iar apoi servite => inserare
    reusita
79  INSERT INTO gateste_serveste
80  VALUES(200, 30, 51, 5);
81
82  INSERT INTO gateste_serveste
83  VALUES(201, 30, 51, 5);
84
85  --incercam sa stergem inregistrarile introduse
86  DELETE FROM gateste_serveste
87  WHERE id_angajat = 200 AND id_comanda = 30 AND id_produs = 51;
88  DELETE FROM gateste_serveste
89  WHERE id_angajat = 201 AND id_comanda = 30 AND id_produs = 51;

```

Exemplu inserare de produse servite înainte de a fi gătite, inserarea nu este efectuată pentru că este declanșat trigger-ul:

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, there are tabs for 'projectSGBD_create.sql', 'projectSGBD_6-9.sql', and 'projectSGBD_10-14.sql'. The main window contains a 'Worksheet' tab with a 'Query Builder' section. The code in the builder is as follows:

```

    INTO can_stearsa
    FROM gateste_serveste
    WHERE id_angajat = :OLD.id_angajat AND id_comanda = :OLD.id_comanda AND id_produs = :OLD.id_produs;

    --rezolvare fara a aparea eroarea 'table is mutating'
    --can_stearsa := :OLD.cantitate;

    DBMS_OUTPUT.PUT_LINE('Cantitatea ' || can_stearsa || ' a fost stearsa cu succes');

END IF;
/
ALTER TRIGGER prod_gatit DISABLE;
ALTER TRIGGER prod_gatit ENABLE;

--incercam servirea unor produse care nu sunt gatite => declansam trigger-ul
INSERT INTO gateste_serveste
VALUES(201, 30, 51, 5);

--cazul in care produsele sunt gatite inainte, iar apoi servite => inserare reusita
INSERT INTO gateste_serveste
VALUES(200, 30, 51, 5);

```

To the right of the worksheet, there is an 'Output' pane titled 'Dbms Output' with the following log:

```

Nr produse gatite:0
Nr produse servite:0
Cantitatea care ar putea fi servita:0

```

At the bottom of the interface, there is a 'Script Output' and 'Query Result' pane. The 'Script Output' pane shows the error message:

```

Error starting at line : 117 in command -
INSERT INTO gateste_serveste
VALUES(201, 30, 51, 5)
Error report -
ORA-20000: Cantitatea mentionata nu poate fi servita
ORA-06512: at "PROIECTSGBD.PROD_GATIT", line 49
ORA-04088: error during execution of trigger 'PROIECTSGBD.PROD_GATIT'

```

Exemplu inserare de produse gătite și apoi servite:

The screenshot shows the Oracle SQL Developer interface. In the top tab bar, there are three tabs: 'projectSGBD_create.sql', 'projectSGBD_6-9.sql', and 'projectSGBD_10-14.sql'. The main area is a 'Worksheet' tab with a 'Query Builder' section containing the following PL/SQL code:

```

    FROM gateste_serveste
    WHERE id_angajat = :OLD.id_angajat AND id_comanda = :OLD.id_comanda AND id_producator = :OLD.id_producator;
    --rezolvare fara a aparea eroarea 'table is mutating'
    --can_stearsa := :OLD.cantitate;
    DBMS_OUTPUT.PUT_LINE('Cantitatea ' || can_stearsa || ' a fost stearsa cu succes');
    END IF;
  END;
/
ALTER TRIGGER prod_gatit DISABLE;
ALTER TRIGGER prod_gatit ENABLE;

--incercam servirea unor produse care nu sunt gătite => declansam trigger-ul
INSERT INTO gateste_serveste
VALUES(201, 30, 51, 5);

--cazul in care produsele sunt gătite înainte, iar apoi servite => inserare reușita
INSERT INTO gateste_serveste
VALUES(200, 30, 51, 5);

INSERT INTO gateste_serveste
VALUES(201, 30, 51, 5);
--incercam sa stergem una dintre inregistrările introduse

```

Below the code, the 'Script Output' tab shows the results of the execution:

```

1 row inserted.

1 row inserted.

```

The 'Dbms Output' tab on the right displays the output of the DBMS_OUTPUT.PUT_LINE statements:

```

Nr produse gătite:5
Nr produse servite:0
Cantitatea care ar putea fi servita:5

```

Inserarea este efectuată cu succes:

A screenshot of the Oracle SQL Developer interface showing the results of a SELECT query:

```

SELECT * FROM gateste_serveste;

```

The results are displayed in a 'Query Result' tab:

ID_ANGAJAT	ID_COMANDA	ID_PRODUS	CANTITATE
24	207	70	55
25	203	80	62
26	205	90	50
27	203	10	58
28	204	100	52
29	205	100	52
30	200	30	51
31	201	30	51

The last two rows (30 and 31) are highlighted in blue, indicating they were inserted.

Exemplu ștergere - varianta în care trigger-ul consultă tabelul de care este asociat, ștergerea nu este efectuată pentru că tabelul este mutating:

```

END IF;
END;
/
ALTER TRIGGER prod_gatit DISABLE;
ALTER TRIGGER prod_gatit ENABLE;

--incercam servirea unor produse care nu sunt gatite => declansam trigger-ul
INSERT INTO gateste_serveste
VALUES(201, 30, 51, 5);

--cazul in care produsele sunt gatite inainte, iar apoi servite => inserare reusita
INSERT INTO gateste_serveste
VALUES(200, 30, 51, 5);

INSERT INTO gateste_serveste
VALUES(201, 30, 51, 5);

--incercam sa stergem inregistrarile introduse
DELETE FROM gateste_serveste
WHERE id_angajat = 200 AND id_comanda = 30 AND id_produc = 51;
DELETE FROM gateste_serveste
WHERE id_angajat = 201 AND id_comanda = 30 AND id_produc = 51;

```

Script Output | Query Result | Task completed in 0.068 seconds

```

Error starting at line : 128 in command -
DELETE FROM gateste_serveste
WHERE id_angajat = 200 AND id_comanda = 30 AND id_produc = 51
Error report -
ORA-04091: table PROJECTSGBD.GATESTE_SERVESTE is mutating, trigger/function may not see it
ORA-06512: at "PROJECTSGBD.PROD_GATIT", line 55
ORA-04088: error during execution of trigger 'PROJECTSGBD.PROD_GATIT'

```

Exemplu ștergere - varianta în care valoarea este preluată cu ajutorul atributului OLD:

```

DEMS_OUTPUT.PUT_LINE('Cantitatea care ar putea fi servita:' || TO_CHAR(nr_prod_gatite - nr_prod_servite));
IF nr_prod_gatite - nr_prod_servite < :NEW.cantitate THEN
    RAISE_APPLICATION_ERROR(-20000, 'Cantitatea mentionata nu poate fi servita');
END IF;
END IF;
END IF;
ELSE
    --incercam sa selectam cantitatea din tabelul in care se produce modificarea => tabel mutating
    /*
    SELECT cantitate
    INTO can_stearsa
    FROM gateste_serveste
    WHERE id_angajat = :OLD.id_angajat AND id_comanda = :OLD.id_comanda AND id_produc = :OLD.id_produc;
    */
    --rezolvare fara a aparea eroarea 'table is mutating'
    can_stearsa := :OLD.cantitate;
    DBMS_OUTPUT.PUT_LINE('Cantitatea ' || can_stearsa || ' a fost stearsa cu succes');
END IF;
END;
/
ALTER TRIGGER prod_gatit DISABLE;
ALTER TRIGGER prod_gatit ENABLE;

--incercam servirea unor produse care nu sunt gatite => declansam trigger-ul

```

Script Output | Query Result | Task completed in 0.075 seconds

```

ORA-06512: at "PROJECTSGBD.PROD_GATIT", line 55
ORA-04088: error during execution of trigger 'PROJECTSGBD.PROD_GATIT'

Trigger PROD_GATIT compiled

```

Stergerea este efectuată cu succes:

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there are three tabs: 'projectSGBD_create.sql', 'projectSGBD_6-9.sql', and 'projectSGBD_10-14.sql'. The 'projectSGBD' connection is selected. The main workspace contains a SQL Worksheet with the following code:

```

DBMS_OUTPUT.PUT_LINE('Cantitatea ' || can_stearsa || ' a fost stearsa cu succes');
END IF;
/
ALTER TRIGGER prod_gatit DISABLE;
ALTER TRIGGER prod_gatit ENABLE;

--incercam servirea unor produse care nu sunt gatite => declansam trigger-ul
INSERT INTO gateste_serveste
VALUES(201, 30, 51, 5);

--cazul in care produsele sunt gatite inainte, iar apoi servite => inserare reusita
INSERT INTO gateste_serveste
VALUES(200, 30, 51, 5);

INSERT INTO gateste_serveste
VALUES(201, 30, 51, 5);

--incercam sa stergem inregistrarile introduse
DELETE FROM gateste_serveste
WHERE id_angajat = 200 AND id_comanda = 30 AND id_producator = 51;
DELETE FROM gateste_serveste
WHERE id_angajat = 201 AND id_comanda = 30 AND id_producator = 51;

```

The last two DELETE statements are highlighted in yellow. The bottom-left pane shows the 'Script Output' tab with the message 'Task completed in 0.035 seconds'. The bottom-right pane shows the 'Query Result' tab with the output:

```

1 row deleted.

1 row deleted.

```

The screenshot shows the Oracle SQL Developer interface with a different window. The top part of the window contains the SQL query:

```

SELECT * FROM gateste_serveste;

```

The bottom part of the window shows the 'Query Result' tab with the message 'All Rows Fetched: 29 in 0.003 seconds'. Below this, a table displays the data from the 'gateste_serveste' table:

ID_ANGAJAT	ID_COMANDA	ID_PRODUS	CANTITATE
18	207	10	56
19	201	20	55
20	205	30	57
21	203	40	61
22	201	50	51
23	205	60	53
24	207	70	55
25	203	80	62
26	205	90	50
27	203	10	58
28	204	100	52
29	205	100	52

12. Trigger de tip LDD

Creați un trigger care permite modificarea schemei numai în timpul săptămânii și de către utilizatorul 'PROIECTSGBD'. În plus, trigger-ul va insera informații despre utilizator, data și comandă în tabelul info_modificari la fiecare încercare de modificare a schemei.

```

1  CREATE TABLE info_modificari --creare tabel info_modificari
2      (nume_user varchar(30),
3       comanda varchar(15),
4       nume_object varchar(30),
5       data_modificare DATE
6      );
7
8  CREATE OR REPLACE TRIGGER modificare_schema
9      BEFORE CREATE OR DROP OR ALTER ON SCHEMA
10     BEGIN
11         INSERT INTO info_modificari --adaugam informatii in tabel
12             VALUES(SYS.LOGIN_USER, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME, SYSDATE);
13
14        IF USER != UPPER('proiectSGBD') THEN --verificam user-ul
15            RAISE_APPLICATION_ERROR(-20010,'Numai utilizatorul proiectSGBD poate
16                modifica schema!');
17        ELSIF TO_CHAR(SYSDATE, 'DAY') = 'SATURDAY' OR TO_CHAR(SYSDATE, 'DAY') =
18            'SUNDAY' THEN --verificam ziua saptamanii
19            RAISE_APPLICATION_ERROR(-20011,'Schema poate fi modificata numai in
20                timpul saptamanii!');
21        END IF;
22    END;
23    /
24
25    --pentru a ilustra inserarea datelor in tabelul info_modificari rulam
26    urmatoarele instructiuni
27    CREATE TABLE incercare
28        (nume varchar(20)
29      );
30
31    ALTER TABLE incercare
32        ADD prenume varchar(20);
33
34    DROP TABLE incercare;

```

```

31
32 ALTER TABLE angajati
33 ADD salariu NUMBER;
34
35 ALTER TABLE angajati
36 DROP COLUMN salariu;

```

Modificările au avut loc cu succes deoarece utilizatorul este 'PROIECTSGBD' și au fost rulate în timpul săptămânii:

```

projectSGBD_create.sql projectSGBD_6-9.sql projectSGBD_10-14.sql
SQL Worksheet History
Worksheet Query Builder
  (nume varchar(20)
  );

ALTER TABLE incercare
ADD prenume varchar(20);

DROP TABLE incercare;

ALTER TABLE angajati
ADD salariu NUMBER;

ALTER TABLE angajati
DROP COLUMN salariu;

```

Script Output | Query Result | Task completed in 0.056 seconds

Table INFO_MODIFICARI created.

Trigger MODIFICARE_SCHEMA compiled

Table INCERCARE created.

Table INCERCARE altered.

Table INCERCARE dropped.

Table ANGAJATI altered.

Table ANGAJATI altered.

Informațiile despre modificări au fost introduse în tabelul info_modificari:

```

SELECT * FROM info_modificari;

```

Script Output | Query Result | All Rows Fetched: 5 in 0.005 seconds

NUME_USER	COMANDA	NUME_OBJECT	DATA_MODIFICARE
1 PROIECTSGBD CREATE	INCERCARE	06-JAN-22	
2 PROIECTSGBD ALTER	INCERCARE	06-JAN-22	
3 PROIECTSGBD DROP	INCERCARE	06-JAN-22	
4 PROIECTSGBD ALTER	ANGAJATI	06-JAN-22	
5 PROIECTSGBD ALTER	ANGAJATI	06-JAN-22	

13. Pachet care conține obiectele definite anterior

Pentru a rezolva cerința am introdus subprogramele definite la exercițiile 6-9 în interiorul unui pachet.

```

1  --specificatia pachetului:
2  CREATE OR REPLACE PACKAGE pachet_proiect AS
3      --functie ex. 6
4      FUNCTION p_comenzi_furnizor
5          (nume_furnizori.denumire%TYPE)
6      RETURN NUMBER;
7
8      --procedura ex. 7
9      PROCEDURE p_produse_gatite
10         (v_nume_angajati.nume%TYPE,
11          v_prenume_angajati.prenume%TYPE);
12
13     --functie ex. 8
14     FUNCTION p_nr_produse
15         (zi DATE,
16          categ categorii.nume_categorie%TYPE)
17     RETURN NUMBER;
18
19     --procedura ex. 9
20     PROCEDURE p_furnizori_evenimente
21         (v_nume_clienti.nume%TYPE,
22          v_prenume_clienti.prenume%TYPE);
23 END pachet_proiect;
24 /
25
26 --corful pachetului:
27 CREATE OR REPLACE PACKAGE BODY pachet_proiect AS
28     --functie ex. 6
29     FUNCTION p_comenzi_furnizor
30         (nume_furnizori.denumire%TYPE)
31     RETURN NUMBER IS
32         TYPE vector IS VARRAY(100) OF ingredientе.id_ingredient%TYPE;
33         --presupunem ca nu avem mai mult de 100 de ingrediente
34         t_ing vector := vector(); --vector folosit pentru a retine lista de
                           --ingrediente furnizate

```

```

35  TYPE tablou_indexat IS TABLE OF produse.id_produs%TYPE INDEX BY
36      PLS_INTEGER;
37  t_prod tablou_indexat; --tablou indexat folosit pentru a retine lista de
38      produse ce contin ingredientele furnizorului
39  t_prod_aux tablou_indexat; --tablou indexat auxiliar folosit pentru a
40      adauga in tabloul indexat mai sus definit numai elemente distincte
41
42
43  TYPE tablou_imbricat IS TABLE OF comenzi.id_comanda%TYPE;
44  t_comenzi tablou_imbricat:= tablou_imbricat(); --tablou imbricat folosit
45      pentru a retine comenziile ce contin produse din t_prod
46  t_comenzi_aux tablou_imbricat:= tablou_imbricat();--tablou imbricat
47      auxiliar folosit pentru a adauga in tabloul imbricat mai sus definit
48      numai elemente distincte
49
50
51  id_ing ingrediente.id_ingredient%TYPE;
52  id_prod produse.id_produs%TYPE;
53  v_id furnizori.id_furnizor%TYPE;
54  aux NUMBER;
55
56 BEGIN
57     SELECT id_furnizor --selectez id-ul furnizorului dat
58     INTO v_id
59     FROM furnizori
60     WHERE upper(denumire) = upper(nume);
61
62     SELECT id_ingredient --introduc in vectorul t_ing id-urile ingredientelor
63     BULK COLLECT INTO t_ing
64     FROM ingrediente
65     WHERE id_furnizor = v_id;
66
67     DBMS_OUTPUT.PUT_LINE('Id furnizor: ' || v_id);
68     DBMS_OUTPUT.PUT_LINE('Ingrediente: ');
69     FOR i IN t_ing.FIRST..t_ing.LAST LOOP --afisare id-uri ingrediente
70         DBMS_OUTPUT.PUT_LINE(t_ing(i));
71     END LOOP;
72
73     id_ing := t_ing(t_ing.FIRST);
74
75     SELECT DISTINCT id_produs --initializez tabloul indexat cu produsele
76         gatite cu primul ingredient
77     BULK COLLECT INTO t_prod
78     FROM contine
79     WHERE id_ingredient = id_ing;
80
81     FOR i IN t_ing.FIRST+1..t_ing.LAST LOOP -- pentru fiecare ingredient
82         ramas
83
84         id_ing := t_ing(i);

```

```

74   SELECT DISTINCT id_produs --introduc in tabloul indexat auxiliar
75     produsele gatite cu ingredientul curent
76   BULK COLLECT INTO t_prod_aux
77   FROM contine
78   WHERE id_ingredient = id_ing;
79
80   --verificare daca exista deja produsul in lista finala
81   FOR k IN t_prod_aux.FIRST..t_prod_aux.LAST LOOP
82     aux := 0;
83
84     FOR j IN t_prod.FIRST..t_prod.LAST LOOP
85       IF t_prod_aux(k) = t_prod(j) THEN
86         aux := 1;
87       END IF;
88     END LOOP;
89
90     --daca aux = 0 inseamna ca produsul nu exista deja in lista, asa
91     --ca il adaugam la final
92     IF aux = 0 THEN
93       t_prod(t_prod.LAST + 1) := t_prod_aux(k);
94     END IF;
95
96   END LOOP;
97
98   DBMS_OUTPUT.PUT_LINE('Produse: ');
99   FOR i IN t_prod.FIRST..t_prod.LAST LOOP --afisare id-uri produse
100     DBMS_OUTPUT.PUT_LINE(t_prod(i));
101   END LOOP;
102
103   id_prod := t_prod(t_prod.FIRST);
104   SELECT DISTINCT id_comanda --initializez tabloul imbricat t_comenzi cu
105     id-ul comenziilor ce contin primul produs din t_prod
106   BULK COLLECT INTO t_comenzi
107   FROM gateste_serveste
108   WHERE id_produs = id_prod;
109
110   FOR i IN t_prod.FIRST+1..t_prod.LAST LOOP
111     id_prod := t_prod(i);
112     SELECT DISTINCT id_comanda --introduc in tabloul imbricat auxiliar
113       id-ul comenziilor ce contin primul produs din t_prod
114     BULK COLLECT INTO t_comenzi_aux
115     FROM gateste_serveste
116     WHERE id_produs = id_prod;
117
118   --verificare daca exista deja comanda in lista finala
119   FOR k IN t_comenzi_aux.FIRST..t_comenzi_aux.LAST LOOP
120

```

```

117         aux := 0;
118
119         FOR j IN t_comenzi.FIRST..t_comenzi.LAST LOOP
120             IF t_comenzi_aux(k) = t_comenzi(j) THEN
121                 aux :=1;
122             END IF;
123         END LOOP;
124
125         --daca aux = 0 inseamna ca id-ul comenzii nu exista deja in
126         --lista, asa ca il adaugam la final
127         IF aux = 0 THEN
128             t_comenzi.EXTEND;
129             t_comenzi(t_comenzi.LAST) := t_comenzi_aux(k);
130             END IF;
131         END LOOP;
132     END LOOP;
133
134     DBMS_OUTPUT.PUT_LINE('Comenzi: ');
135     FOR i IN t_comenzi.FIRST..t_comenzi.LAST LOOP --afisare id-uri comenzii
136         DBMS_OUTPUT.PUT_LINE(t_comenzi(i));
137     END LOOP;
138
139     RETURN t_comenzi.COUNT; --returnare numar de elemente ale tabloului
140     imbricat cu id-urile comenzilor
141
142     EXCEPTION
143         WHEN NO_DATA_FOUND THEN --cazul in care denumirea data nu corespunde
144             cu denumirea unui furnizor
145             RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun furnizor cu
146             denumirea data');
147         WHEN TOO_MANY_ROWS THEN --cazul in care exista mai multi furnizori cu
148             denumirea introdusa
149             RAISE_APPLICATION_ERROR(-20001, 'Există mai multi furnizori cu
150             denumirea data');
151     END p_comenzi_furnizor;
152
153     --procedura ex. 7
154     PROCEDURE p_produse_gatite
155         (v_nume_angajati.nume%TYPE,
156          v_prenume_angajati.prenume%TYPE)
157     IS
158         v_id_angajati.id_angajat%TYPE;
159         v_produse.id_produs%TYPE;
160         nume_prod produse.denumire%TYPE;
161         --cursor clasic
162         CURSOR c_prod IS
163             SELECT DISTINCT id_produs

```

```

158   FROM gateste_serveste
159 WHERE id_angajat = (SELECT id_angajat
160   FROM angajati
161   WHERE upper(nume)=upper(v_nume) AND
162   upper(prenume)=upper(v_prenume)
163   AND tip_angajat = 'bucatar'
164   );
165 --ciclu cursor parametrizat
166 CURSOR c_ingr(id_prod produse.id_produs%TYPE) IS
167   SELECT denumire
168   FROM ingrediente
169   WHERE id_ingredient IN (SELECT id_ingredient
170   FROM contine
171   WHERE id_produs = id_prod
172   );
173 BEGIN
174   SELECT id_angajat --selectare id bucatar
175   INTO v_id
176   FROM angajati
177   WHERE upper(nume)=upper(v_nume) AND upper(prenume)=upper(v_prenume) AND
178     tip_angajat = 'bucatar';
179   --am pus si conditia ca numele introdus sa fie al unui bucatar
180
181   DBMS_OUTPUT.PUT_LINE('BUCATARUL ' || initcap(v_nume) || ' ' ||
182                         initcap(v_prenume));
183
184   OPEN c_prod; --deschidere cursor
185   LOOP FETCH c_prod INTO v_prod; --incarcare linii pe rand in variabila
186     v_prod
187   EXIT WHEN c_prod%NOTFOUND;
188   SELECT denumire --selectare denumire produs
189   INTO nume_prod
190   FROM produse
191   WHERE id_produs = v_prod;
192   DBMS_OUTPUT.PUT_LINE('Produsul: ' || nume_prod);
193   FOR ingr IN c_ingr(v_prod) LOOP --ciclu cursor pentru afisarea
194     ingredientelor
195     DBMS_OUTPUT.PUT_LINE(' -> ' || ingr.denumire);
196   END LOOP;
197   DBMS_OUTPUT.NEW_LINE;
198 END LOOP;

199
200 --daca numarul liniilor este 0, atunci inseamna ca bucatarul nu a gatit
201   nimic inca
202 IF c_prod%ROWCOUNT = 0 THEN
203   DBMS_OUTPUT.PUT_LINE('Bucatarul dat nu a gatit niciun produs pana
204   acum');

```

```

198    END IF;
199    CLOSE c_prod; --inchidere cursor
200
201    EXCEPTION
202        WHEN NO_DATA_FOUND THEN --cazul in care nu exista niciun bucatar cu
203            numele dat
204            RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun bucatar cu
205                numele dat');
206        WHEN TOO_MANY_ROWS THEN --cazul in care exista mai multi bucatari cu
207            numele dat
208            RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi bucatari cu
209                numele dat');
210    END p_produse_gatite;
211
212    --functie ex. 8
213    FUNCTION p_nr_produse
214        (zi DATE,
215         categ categorii.nume_categorie%TYPE)
216    RETURN NUMBER IS
217        nr NUMBER := 0;
218        nr_aux NUMBER;
219        TYPE t_ind IS TABLE OF produse.id_produs%TYPE
220            INDEX BY PLS_INTEGER;
221        v_ind t_ind; --tablou indexat pentru a retine produsele ce fac parte din
222            categoria data
223        id_ev evenimente.id_eveniment%TYPE;
224        exc_categ EXCEPTION;
225        exc_produse EXCEPTION;
226
227    BEGIN
228        --verific daca exista categoria
229        SELECT COUNT(id_categorie)
230        INTO nr
231        FROM categorii
232        WHERE upper(nume_categorie) = upper(categ);
233
234        IF nr = 0 THEN --cazul in care categoria data nu exista
235            RAISE exc_categ;
236        END IF;
237
238        --verific daca exista produse in categoria data
239        SELECT COUNT(id_produs)
240        INTO nr
241        FROM produse p JOIN categorii c ON (p.id_categorie=c.id_categorie)
242        WHERE upper(nume_categorie) = upper(categ);
243
244        IF nr = 0 THEN --cazul in care nu exista produse in categoria data
245            RAISE exc_produse;

```

```

240    END IF;
241
242    --selectez id-urile produselor din categoria data
243    SELECT id_produs
244        BULK COLLECT INTO v_ind
245        FROM produse p JOIN categorii c ON (p.id_categorie=c.id_categorie)
246        WHERE upper(nume_categorie) = upper(categ);
247
248    --selectez id-ul evenimentului din ziua data
249    SELECT id_eveniment
250        INTO id_ev
251        FROM evenimente
252        WHERE data_eveniment = zi;
253
254    nr := 0;
255    FOR i IN v_ind.FIRST..v_ind.LAST LOOP
256        SELECT SUM(cantitate) --retin cantitatea vanduta pentru fiecare
257            produs in parte
258            INTO nr_aux
259            FROM gateste_serveste g JOIN comenzi c ON (g.id_comanda =
260                c.id_comanda)
261                    JOIN angajati a ON (g.id_angajat = a.id_angajat)
262                    WHERE c.id_eveniment = id_ev AND tip_angajat = 'ospatar'
263                        AND id_produs = v_ind(i);
264
265        nr := nr + NVL(nr_aux, 0); -- adaug cantitatea la suma totala
266    END LOOP;
267
268    RETURN nr; --returnez numarul de produse obtinut
269
270    EXCEPTION
271        WHEN NO_DATA_FOUND THEN --cazul in care nu exista un eveniment in
272            ziua data
273                RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun eveniment in
274                    ziua data');
275        WHEN TOO_MANY_ROWS THEN --cazul in care exista mai multe evenimente
276            in ziua data
277                RAISE_APPLICATION_ERROR(-20001, 'Exista mai multe evenimente in
278                    ziua data');
279        WHEN exc_categ THEN --cazul in care categoria introdusa nu exista sau
280            cazul in care inca nu sunt introduse produsele care fac parte din
281            aceasta
282                RAISE_APPLICATION_ERROR(-20002, 'Categoria data nu exista');
283        WHEN exc_produse THEN --cazul in care inca nu sunt introduse produse
284            care sa faca parte din categoria data
285                RAISE_APPLICATION_ERROR(-20003, 'Nu exista produse care sa faca
286                    parte din categoria data');

```

```

277    END p_nr_produse;

278

279    --procedura ex. 9
280    PROCEDURE p_furnizori_evenimente
281        (v_nume clienti.nume%TYPE,
282         v_prenume clienti.prenume%TYPE)
283    IS
284        --cursor imbricat
285        CURSOR ev(id_c clienti.id_client%TYPE) IS
286            SELECT data_eveniment, denumire_tip, --selectez date despre
287                  eveniment
288                  CURSOR(SELECT DISTINCT f.denumire --selectez
289                          furnizorii ceruti in functie de eveniment
290                          FROM furnizori f JOIN ingrediente i ON
291                              (f.id_furnizor=i.id_furnizor)
292                          JOIN contine c ON
293                              (i.id_ingredient=c.id_ingredient)
294                          JOIN produse p ON
295                              (p.id_produs=c.id_produs)
296                          JOIN gateste_serveste g ON
297                              (p.id_produs=g.id_produs)
298                          WHERE g.id_comanda = c.id_comanda
299                      )
300            FROM evenimente e JOIN tipuri_evenimente t ON
301                (e.id_tip_eveniment = t.id_tip_eveniment)
302                JOIN comenzi c ON (e.id_eveniment =
303                    c.id_eveniment)
304                JOIN clienti cli ON (cli.id_client =
305                    c.id_client)
306                WHERE cli.id_client = id_c;

307        c_furnizori SYS_REFCURSOR;
308        v_id clienti.id_client%TYPE;
309        data_ev evenimente.data_eveniment%TYPE;
310        nume_tip tipuri_evenimente.denumire_tip%TYPE;
311        TYPE tablou_indexat IS TABLE OF furnizori.denumire%TYPE INDEX BY
312            PLS_INTEGER;
313        t_furnizori tablou_indexat; --tablou indexat pt a retine denumirea
314            furnizorilor
315        exc EXCEPTION;
316    BEGIN
317        SELECT id_client --selectez id-ul clientului introdus
318        INTO v_id
319        FROM clienti
320        WHERE upper(nume)=upper(v_nume) AND upper(prenume)=upper(v_prenume);

321        OPEN ev(v_id); --deschid cursorul

```

```

313
314     LOOP
315         FETCH ev INTO data_ev, nume_tip, c_furnizori;
316         EXIT WHEN ev%NOTFOUND;
317
318         DBMS_OUTPUT.PUT_LINE('Eveniment: ' || nume_tip || ' din data ' ||
319             TO_CHAR(data_ev));
320         FETCH c_furnizori BULK COLLECT INTO t_furnizori;
321
322         IF t_furnizori.COUNT = 0 THEN
323             DBMS_OUTPUT.PUT_LINE('Pentru acest eveniment nu au fost date
324                 comenzi');
325         ELSE
326             FOR i IN t_furnizori.FIRST..t_furnizori.LAST LOOP
327                 DBMS_OUTPUT.PUT_LINE(i|| '.' || t_furnizori(i)); --afisez
328                     furnizori
329             END LOOP;
330         END IF;
331
332         DBMS_OUTPUT.NEW_LINE;
333     END LOOP;
334
335     IF ev%ROWCOUNT = 0 THEN --cazul in care clientul nu a dat nicio comanda
336         pentru un eveniment
337         RAISE exc;
338     END IF;
339
340     CLOSE ev;
341     EXCEPTION
342         WHEN NO_DATA_FOUND THEN -- cazul in care nu exista niciun client cu
343             numele dat
344             RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun client cu
345                 numele dat');
346         WHEN TOO_MANY_ROWS THEN -- cazul in care exista mai multi clienti cu
347             numele dat
348             RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi clienti cu
349                 numele dat');
350         WHEN exc THEN -- cazul in care clientul nu a dat nicio comanda pentru
351             un eveniment
352             RAISE_APPLICATION_ERROR(-20002, 'Clientul dat nu a dat nicio
353                 comanda pentru un eveniment');
354     END p_furnizori_evenimente;
355 END pachet_proiect;
356 /
357
358 BEGIN
359     --apelare functie ex 6 - din pachet:

```

```

350     DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' ||
351                         pachet_proiect.p_comenzi_furnizor('SC Delicios'));
351
352 /
353
353 BEGIN
354     --apelare procedura ex 7 - din pachet:
355     pachet_proiect.p_produse_gatite('Vilcu', 'Elena');
356
357 /
358
358 BEGIN
359     --apelare functie ex 8 - din pachet:
360     DBMS_OUTPUT.PUT_LINE('Au fost comandate ' ||
361                         pachet_proiect.p_nr_produse(TO_DATE('03-APR-21'), 'friptura') || ,
362                         produse din aceasta categorie');
361
362 /
363
363 BEGIN
364     --apelare procedura ex 9 - din pachet:
365     pachet_proiect.p_furnizori_evenimente('Popescu', 'Maria');
366
367 /

```

Exemple apelare subprograme din pachet:

The screenshot shows the Oracle SQL Developer interface with several tabs open. The 'Script Output' tab at the bottom displays the completed procedure:

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

The 'Dbms Output' tab on the right shows the execution results:

```

Id furnizor: 1040
Ingrediente:
177
178
185
Produse:
53
59
Comenzi:
60
30
70
10
Numarul de comenzi: 4

```

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

- Top Bar:** Shows three tabs: projectSGBD_create.sql, projectSGBD_6-9.sql, and projectSGBD_10-14.sql.
- Toolbar:** Standard SQL Worksheet toolbar.
- Worksheet:** Contains a PL/SQL procedure named `pachet_proiect`. The code includes several `BEGIN` blocks, each containing DBMS_OUTPUT.PUT_LINE statements. One block specifically calls `pachet_proiect.p_produse_gatite('Vilcu', 'Elena')`.
- Dbms Output:** A window titled "projectSGBD" showing the output of the DBMS_OUTPUT.PUT_LINE statements. It displays:
 - BUCATARUL Vilcu Elena
 - Produsul: clatite
 - > faina
 - > ciocolata
 - Produsul: paste carbonara
 - > spaghetti
 - > parmezan
- Script Output:** Shows the message "PL/SQL procedure successfully completed."

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

- Top Bar:** Shows three tabs: projectSGBD_create.sql, projectSGBD_6-9.sql, and projectSGBD_10-14.sql.
- Toolbar:** Standard SQL Worksheet toolbar.
- Worksheet:** Contains the same PL/SQL procedure `pachet_proiect` as the previous screenshot.
- Dbms Output:** A window titled "projectSGBD" showing the output of the DBMS_OUTPUT.PUT_LINE statements. It displays the message: "Au fost comandate 20 produse din aceasta categorie".
- Script Output:** Shows the message "PL/SQL procedure successfully completed."

The screenshot shows the Oracle SQL Developer interface during the execution of a PL/SQL procedure. The main window displays the code of the procedure, which includes several BEGIN blocks calling different functions and procedures from a package named 'pachet_proiect'. The code is annotated with comments explaining the purpose of each block.

```

projectSGBD_create.sql  projectSGBD_6-9.sql  projectSGBD_10-14.sql
SQL Worksheet History
Query Builder
Worksheet Query Builder
--EXC THEN    cauză în cazul în care clientul nu a dat nicio comandă pentru un eveniment
      RAISE_APPLICATION_ERROR(-20002, 'Clientul dat nu a dat nicio comandă pentru un eveniment');
END p_furnizori_evenimente;
END pachet_proiect;
/
BEGIN
  --apelare functie ex 6 - din pachet:
  DBMS_OUTPUT.PUT_LINE('Numarul de comenzi: ' || pachet_proiect.p_comenzi_furnizor('SC Delicios'));
END;
/
BEGIN
  --apelare procedura ex 7 - din pachet:
  pachet_proiect.p_produse_gatite('Vilcu', 'Elena');
END;
/
BEGIN
  --apelare functie ex 8 - din pachet:
  DBMS_OUTPUT.PUT_LINE('Au fost comandate ' || pachet_proiect.p_nr_produse(TO_DATE('03-APR-21'), 'f'));
END;
/
BEGIN
  --apelare procedura ex 9 - din pachet:
  pachet_proiect.p_furnizori_evenimente('Popescu', 'Maria');
END;
/

```

The 'Dbms Output' window on the right shows the results of the DBMS_OUTPUT.PUT_LINE statements. It lists three events: 'revedere' on 15-SEP-20, 'cununie' on 07-OCT-21, and another event with the same details as the first one.

Script Output X | Task completed in 0.047 seconds

PL/SQL procedure successfully completed.

14. Pachet care include tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate

Definiți un pachet care prin intermediul mai multor subprograme să realizeze statistica încasărilor și produselor comandate pe o perioadă de timp.

Pachetul va conține:

- o funcție care pentru un an dat ca parametru returnează suma încasărilor restaurantului în acel an
- o funcție care returnează unul dintre cele mai bine vândute produse dintr-un an
- o funcție care primește ca parametrii denumirea unui produs și un an și returnează o lista cu id-urile bucătarilor care au gătit produsul în anul dat
- o procedură care primește ca parametrii un an și un nr n și afișează un top n cele mai bine vândute produse în acel an
- o procedură care primește ca parametrii un an și un nr n și afișează suma încasărilor, numărul produselor comandate din fiecare categorie și un top n al produselor din acel an (folosind subprogramele definite anterior)
- o procedură care primește ca parametrii denumirea unui produs și un an și afișează suma încasărilor obținute în urma vânzării produsului în anul dat și ce procent reprezintă aceasta din suma totală obținută, cât și o listă a bucătarilor care au gătit acest produs
- o procedură care primește un interval de ani ca parametru și afișează pentru fiecare an din acest interval o analiză detaliată a anului (suma încasărilor, numărul de produse comandate din fiecare categorie, top 3 cele mai bine vândute produse) și apoi o analiză a uneia dintre cele mai bine vândute produse în acel an (încasări din vânzarea acestuia, procent încasări din totalul încasărilor, lista bucătarilor care au gătit acest produs) (folosind subprogramele definite anterior)

```

1 CREATE OR REPLACE PACKAGE statistica AS
2   TYPE tablou_imb_prod IS TABLE OF produse.id_produs%TYPE;
3   TYPE vector_bucatari IS VARRAY(50) OF angajati.id_angajat%TYPE;
4
5   -- functie care pentru un an dat ca parametru returneaza suma incasarilor
6   -- restaurantului in acel an
7   FUNCTION incasari_an(an NUMBER)
8     RETURN NUMBER;
9
10  -- functie care returneaza unul dintre cele mai bine vandute produse
11  -- dintr-un an
12  FUNCTION produs_top1(an NUMBER)

```

```

11      RETURN produse.denumire%TYPE;
12
13      -- functie care primeste ca parametrii denumirea unui produs si un an si
14          -- returneaza o lista cu id-urile bucatarilor
15      --care au gatit produsul in anul dat
16      FUNCTION bucatari_pentru(prod produse.denumire%TYPE,
17                                an NUMBER)
18          RETURN vector_bucatari;
19
20      -- procedura care primeste ca parametrii un an si un nr n si afiseaza un top
21          -- n cele mai bine vandute produse in acel an
22      PROCEDURE produse_top_n(an NUMBER,
23                               n_top NUMBER);
24
25      -- o procedura care primeste ca parametrii un an si un nr n si afiseaza suma
26          -- incasarilor, nr produselor comandate din fiecare categorie
27      -- si un top n al produselor din acel an (folosind subprogramele definite
28          -- anterior)
29      PROCEDURE analiza_an(an NUMBER,
30                           n NUMBER);
31
32      -- procedura care primeste ca parametrii denumirea unui produs si un an si
33          -- afiseaza suma incasarilor obtinute in urma vanzarii
34      -- produsului in anul dat si ce procent reprezinta aceasta din suma totala
35          -- obtinuta, cat si o lista a bucatarilor care au gatit acest produs
36      PROCEDURE analiza_produs(prod produse.denumire%TYPE,
37                                an NUMBER);
38
39      -- procedura care primeste un interval de ani ca parametru si afiseaza
40          -- pentru fiecare an din acest interval o analiza detaliata a anului
41          -- (suma incasarilor, numarul de produse comandate din fiecare categorie, top
42              -- 3 cele mai bine vandute produse) si apoi o analiza
43          -- a uneia dintre cele mai bine vandute produse in acel an (incasari din
44              -- vanzarea acestuia, procent incasari din totalul incasarilor,
45              -- lista bucatarilor care au gatit acest produs)
46      PROCEDURE analiza_interval (an_s NUMBER,
47                                  an_e NUMBER);
48
49  END statistica;
50 /
51 CREATE OR REPLACE PACKAGE BODY statistica AS
52     -- functie care pentru un an dat ca parametru returneaza suma incasarilor
53         -- restaurantului in acel an
54     FUNCTION incasari_an(an NUMBER)
55         RETURN NUMBER
56     IS
57         incasari NUMBER;
58     BEGIN

```

```

48    SELECT SUM(cantitate * pret) --calculam incasarile ca fiind suma dintre
49        cantitatea de produse * pret
50    INTO incasari
51    FROM gateste_serveste g JOIN produse p ON (g.id_produs = p.id_produs)
52                JOIN comenzi c ON (g.id_comanda = c.id_comanda)
53                JOIN angajati a ON (g.id_angajat = a.id_angajat)
54    WHERE tip_angajat='ospatar'
55    GROUP BY TO_CHAR(data_comanda, 'YYYY')
56    HAVING TO_CHAR(data_comanda, 'YYYY') = an;
57
58    RETURN incasari; --returnam suma calculata
59
60    EXCEPTION
61        WHEN no_data_found THEN --inseamna ca nu exista comenzi in acel an,
62            deci incasarile sunt 0
63            RETURN 0;
64    END incasari_an;
65
66    -- functie care returneaza unul dintre cele mai bine vandute produse
67    -- dintr-un an
68    FUNCTION produs_top1(an NUMBER)
69        RETURN produse.denumire%TYPE
70    IS
71        nr_prod_max NUMBER;
72        v_produse tablou_imb_prod := tablou_imb_prod();
73        nume_prod produse.denumire%TYPE;
74        id_max produse.id_produs%TYPE;
75    BEGIN
76        SELECT MAX(SUM(cantitate)) --selectez cantitatea maxima de produse
77            vandute
78        INTO nr_prod_max
79        FROM gateste_serveste g JOIN produse p ON (g.id_produs = p.id_produs)
80                JOIN comenzi c ON (g.id_comanda = c.id_comanda)
81                JOIN angajati a ON (g.id_angajat = a.id_angajat)
82        WHERE tip_angajat='ospatar' AND TO_CHAR(data_comanda, 'YYYY') = an
83        GROUP BY g.id_produs;
84
85        SELECT g.id_produs --introduc in tabloul imbricat v_produse toate
86            produsele din care s-a vandut
87            -- o cantitate egala cu cantitatea maxima calculata mai
88            sus
89        BULK COLLECT INTO v_produse
90        FROM gateste_serveste g JOIN produse p ON (g.id_produs = p.id_produs)
91                JOIN comenzi c ON (g.id_comanda = c.id_comanda)
92                JOIN angajati a ON (g.id_angajat = a.id_angajat)
93        WHERE tip_angajat='ospatar' AND TO_CHAR(data_comanda, 'YYYY') = an
94        GROUP BY g.id_produs

```

```

89      HAVING SUM(cantitate) = nr_prod_max;
90
91      IF v_produse.COUNT > 0 THEN
92          --pentru ca ne trebuie un singur produs dintre cele mai bine vandute,
93          --luam numai primul produs, cel de pe prima pozitie in tablou
94          id_max := v_produse(v_produse.FIRST);
95          SELECT denumire
96              INTO nume_prod
97              FROM produse
98              WHERE id_produs = id_max;
99
100         RETURN nume_prod;
101     ELSE
102         RETURN '0'; --cazul in care in acel an nu a fost plasata nicio comanda
103     END IF;
104 END produs_top1;
105
106 -- functie care primeste ca parametrii denumirea unui produs si un an si
107 -- returneaza o lista cu id-urile bucatarilor
108 --care au gatit produsul in anul dat
109 FUNCTION bucatari_pentru(prod produse.denumire%TYPE, an NUMBER)
110     RETURN vector_bucatari
111 IS
112     vec_bucatari vector_bucatari := vector_bucatari();
113     id_prod produse.id_produs%TYPE;
114 BEGIN
115     SELECT id_produs --selectez id-ul produsului cu denumirea data
116         INTO id_prod
117         FROM produse
118         WHERE upper(denumire) = upper(prod);
119
120     SELECT DISTINCT g.id_angajat --introduc in vector id-urile bucatarilor
121         care au gatit produsul dat
122     BULK COLLECT INTO vec_bucatari
123     FROM angajati a JOIN gateste_serveste g ON (a.id_angajat = g.id_angajat)
124             JOIN comenzi c ON (g.id_comanda = c.id_comanda)
125     WHERE tip_angajat = 'bucatar' AND TO_CHAR(data_comanda, 'YYYY') = an
126             AND g.id_produs = id_prod;
127
128     RETURN vec_bucatari; --returnez vectorul obtinut
129
130     EXCEPTION
131         WHEN no_data_found THEN -- cazul in care nu exista niciun produs cu
132             denumirea data
133             RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun produs cu
134                 denumirea data');
135         WHEN TOO_MANY_ROWS THEN --cazul in care exista mai multe produse cu

```

```

        denumirea data
131      RAISE_APPLICATION_ERROR(-20001, 'Exista mai multe produse cu
           denumirea data');
132
END bucatari_pentru;

133
-- procedura care primeste ca parametru un an si un nr n si afiseaza un top
  n cele mai bine vandute produse in acel an
134
PROCEDURE produse_top_n(an NUMBER,
                         n_top NUMBER)
135
IS
136
    nr_top NUMBER := 0;
137
    trecere NUMBER := 0;
138
    cant_anterioara NUMBER;
139
    nume_produse.denumire%TYPE;
140
    cantitate_gateste_serveste.cantitate%TYPE;
141
    aux NUMBER := 0;
142
143
CURSOR c_prod --cursor pentru a procesa denumirea produselor alaturi de
               cantitate in ordine descrescatoare a cantitatii vandute
144
IS
145
    SELECT MAX(denumire), SUM(cantitate)
146
        FROM produse p JOIN gateste_serveste g ON (p.id_produs = g.id_produs)
147
              JOIN comenzi c ON (g.id_comanda = c.id_comanda)
148
              JOIN angajati a ON (g.id_angajat = a.id_angajat)
149
        WHERE tip_angajat='ospatar' AND TO_CHAR(data_comanda, 'YYYY') = an
150
        GROUP BY p.id_produs
151
        ORDER BY SUM(cantitate) DESC;
152
153
BEGIN
154
    DBMS_OUTPUT.PUT_LINE('Top ' || n_top || ' produse:');
155
    OPEN c_prod; --deschidem cursorul
156
    LOOP
157
        FETCH c_prod INTO nume, cantitate;
158
        EXIT WHEN c_prod%NOTFOUND OR nr_top = n_top; --ne oprim cand nu mai
               sunt produse de procesat sau cand deja avem top n
159
        trecere := trecere + 1;
160
        IF trecere = 1 THEN
161
            cant_anterioara := cantitate; --la inceput cantitatea anterioara
               devine cantitatea primului produs
162
            nr_top := nr_top + 1;
163
        END IF;
164
165
        IF cant_anterioara <> cantitate THEN --crestem o pozitie in top cand
               gasim o noua cantitate
166
            nr_top := nr_top + 1;
167
            cant_anterioara := cantitate;
168
        aux := 1;
169
    END IF;
170

```

```

171
172     DBMS_OUTPUT.PUT_LINE(nr_top || ', ' || nume);
173 END LOOP;
174 CLOSE c_prod;
175 FOR i IN nr_top+1..n_top LOOP -- pentru pozitiile din top ramase
176     necompletate afisam o -
177     DBMS_OUTPUT.PUT_LINE(i || ', -');
178 END LOOP;
179 END produse_top_n;

180 -- o procedura care primeste ca parametru un an si un nr n si afiseaza suma
181     incasarilor, nr produselor comandate din fiecare categorie
182 --si un top n al produselor din acel an (folosind subprogramele definite
183     anterior)
184 PROCEDURE analiza_an(an NUMBER,
185                         n NUMBER)
186 IS
187     nr NUMBER := 0;
188 BEGIN
189     DBMS_OUTPUT.PUT_LINE('Suma incasari: ' || incasari_an(an)); --afisam
190         incasarile din anul dat
191     DBMS_OUTPUT.NEW_LINE;
192     DBMS_OUTPUT.PUT_LINE('Nr de produse comandate din ');

193     FOR categ IN (SELECT nume_categoria
194                     FROM categorii) LOOP --pentru fiecare categorie
195         SELECT SUM(cantitate) --calculez nr de produse comandate
196             INTO nr
197             FROM produse p JOIN gateste_serveste g ON (p.id_produs = g.id_produs)
198                             JOIN comenzi c ON (g.id_comanda = c.id_comanda)
199                             JOIN angajati a ON (g.id_angajat = a.id_angajat)
200                             JOIN categorii cat ON (p.id_categoria =
201                                         cat.id_categoria)
202 WHERE tip_angajat='ospatar' AND TO_CHAR(data_comanda, 'YYYY') = an
203         AND nume_categoria = categ.nume_categoria;

204     DBMS_OUTPUT.PUT_LINE('- categoria ' || categ.nume_categoria || ' -> '
205         || NVL(nr, 0) || ' produse');

206     END LOOP;
207     DBMS_OUTPUT.NEW_LINE;
208     produse_top_n(an, n); --apelam procedura pentru a afisa top n produse
209     DBMS_OUTPUT.NEW_LINE;
210 END analiza_an;

211 -- procedura care primeste ca parametru denumirea unui produs si un an si
212     afiseaza suma incasarilor obtinute in urma vanzarii

```

```

211    --produsului in anul dat si ce procent reprezinta aceasta din suma totala
212        obtinuta, cat si o lista a bucatarilor care au gatit acest produs
213    PROCEDURE analiza_produs(prod produse.denumire%TYPE,
214                                an NUMBER)
215    IS
216        incasari_totale NUMBER := 0;
217        incasari_produs NUMBER := 0;
218        an_curent NUMBER;
219        procent NUMBER;
220        id_prod produse.id_produs%TYPE;
221        vec_bucatari vector_bucatari();
222    BEGIN
223        SELECT id_produs --selectez id-ul produsului cu denumirea data
224            INTO id_prod
225            FROM produse
226            WHERE upper(denumire) = upper(prod);
227
228        incasari_totale := incasari_an(an);
229
230        SELECT NVL(SUM(cantitate * pret), 0) --calculam incasarile obtinute in
231            urma vanzarii produsului dat
232            INTO incasari_produs
233            FROM gateste_serveste g JOIN produse p ON (g.id_produs = p.id_produs)
234                JOIN comenzi c ON (g.id_comanda = c.id_comanda)
235                JOIN angajati a ON (g.id_angajat = a.id_angajat)
236            WHERE tip_angajat='ospatar' AND g.id_produs = id_prod
237                AND TO_CHAR(data_comanda, 'YYYY') = an;
238
239        DBMS_OUTPUT.PUT_LINE('Incasari din urma vanzarii produsului: ' ||
240                            incasari_produs);
241
242        procent := TRUNC((incasari_produs * 100) / incasari_totale, 2);
243        --calculam procentul
244
245        DBMS_OUTPUT.PUT_LINE('Procent din suma totala a incasarilor: ' ||
246                            procent || '%');
247        DBMS_OUTPUT.NEW_LINE;
248
249        vec_bucatari := bucatari_pentru(prod, an);
250        IF vec_bucatari.COUNT > 0 THEN
251            DBMS_OUTPUT.PUT_LINE('Bucatarii care au gatit acest produs: ');
252            --afisam si bucatarii care au gatit acest produs
253            FOR i IN vec_bucatari.FIRST..vec_bucatari.LAST LOOP
254                DBMS_OUTPUT.PUT_LINE(i || ' . ' || vec_bucatari(i));
255            END LOOP;
256        ELSE
257            DBMS_OUTPUT.PUT_LINE('Acest produs nu este gatit.');//face parte din

```

```

        categoria bauturi
252    END IF;
253    vec_bucatari.delete;

254
255    EXCEPTION
256        WHEN no_data_found THEN -- cazul in care nu exista niciun produs cu
257            denumirea data
258            RAISE_APPLICATION_ERROR(-20000, 'Nu exista niciun produs cu
259                denumirea data');
260        WHEN TOO_MANY_ROWS THEN --cazul in care exista mai multe produse cu
261            denumirea data
262            RAISE_APPLICATION_ERROR(-20001, 'Există mai multe produse cu
263                denumirea data');

264    END analiza_producător;

265
266    -- procedura care primește un interval de ani ca parametru și afisează
267    -- pentru fiecare an din acest interval o analiză detaliată a anului
268    --(suma incasarilor, numărul de produse comandate din fiecare categorie, top
269    -- 3 cele mai bine vândute produse) și apoi o analiză
270    -- a uneia dintre cele mai bine vândute produse în acel an (incasări din
271    -- vânzarea acestuia, procent incasări din totalul incasarilor,
272    -- lista bucătărilor care au găsit acest produs)
273    PROCEDURE analiza_interval (an_s NUMBER,
274                                an_e NUMBER)
275    IS
276        exc_interval EXCEPTION;
277        an_aux NUMBER;
278        an_curent NUMBER;
279        an_start NUMBER;
280        an_end NUMBER;
281        produs1 produse.denumire%TYPE;
282
283    BEGIN
284        an_start := an_s;
285        an_end := an_e;
286        an_curent := TO_NUMBER(TO_CHAR(SYSDATE, 'YYYY'));
287
288        --verificam intervalul dat
289        IF an_start > an_end THEN --daca anii nu sunt in ordine crescatoare ii
290            interschimbam
291            an_aux := an_start;
292            an_start := an_end;
293            an_end := an_aux;
294        END IF;
295
296        IF an_start < 2019 THEN
297            an_start := 2019; --pornim cu anul 2019 daca anul de start dat este
298                mai mic decat acesta

```

```

289     IF an_end < 2019 THEN --daca ambii ani sunt mai mici decat 2019,
290         aruncam o exceptie
291         RAISE exc_interval;
292     END IF;
293     DBMS_OUTPUT.PUT_LINE('Restaurantul a fost infiintat in anul 2019.
294                           Analiza afisata va incepe cu acest an.');
295 END IF;
296
297 IF an_end > an_curent THEN
298     an_end := an_curent; --incheiem cu anul curent daca anul de incheiere
299     dat este mai mare
300     IF an_start > an_curent THEN --daca ambii ani sunt mai mari decat
301         anul curent, aruncam o exceptie
302         RAISE exc_interval;
303     END IF;
304     DBMS_OUTPUT.PUT_LINE('Nu poate fi afisata analiza pe anii urmatori.
305                           Analiza se va termina cu anul curent.');
306 END IF;
307
308 --afisare analiza pentru fiecare an
309 FOR i IN an_start..an_end LOOP
310     DBMS_OUTPUT.PUT_LINE('-----');
311     DBMS_OUTPUT.PUT_LINE('ANUL ' || i);
312
313     analiza_an(i, 3); --apelam procedura definita anterior pentru analiza
314     anului
315
316     produs1 := produs_top1(i);
317     IF produs1 != '0' THEN
318         DBMS_OUTPUT.PUT_LINE('Cel mai vandut produs: ' || produs1);
319         --afisam si unul dintre cele mai bine vandute produse
320         DBMS_OUTPUT.NEW_LINE;
321         analiza_produc(produs1, i); --apelam procedura definita anterior
322         pentru analiza produsului
323     ELSE
324         DBMS_OUTPUT.PUT_LINE('Nu au fost plasate comenzi in acest an.');
325     END IF;
326     DBMS_OUTPUT.NEW_LINE;
327 END LOOP;
328
329 EXCEPTION
330     WHEN exc_interval THEN
331         RAISE_APPLICATION_ERROR(-20000, 'Interval invalid! Introduceti un
332                               interval intre 2019 si anul curent.');
333     END analiza_interval;
334 END statistica;
335 /

```

```

327
328 BEGIN
329     statistica.analiza_interval(2017, 2023);
330     --statistica.analiza_interval(2017, 2018); --exceptie: interval invalid
331 END;
332 /
333 --pentru a introduce cei doi ani de la tastatura:
334 DECLARE
335     an_start NUMBER := &p_an_start;
336     an_end NUMBER := &p_an_end;
337 BEGIN
338     statistica.analiza_interval(an_start, an_end);
339 END;

```

Exemplu apelare procedură analiza_interval din pachet cu un interval valid:

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, several tabs are visible: projectSGBD_create.sql, projectSGBD_6-9.sql, projectSGBD_10-14.sql, projectSGBD_ex14.sql, and Welcome... The active tab is projectSGBD.

In the central area, there are two panes. The left pane, titled "Worksheet", contains PL/SQL code. The right pane, titled "Dbms Output", displays the results of the executed code.

Worksheet Content:

```

    produs1 := produs_top1(i);
    IF produs1 != '0' THEN
        DBMS_OUTPUT.PUT_LINE('Cel mai vandut produs: ' || produs1);
        DBMS_OUTPUT.NEW_LINE;
        analiza_produc(produs1, i); --apelam procedura definita anterior
    ELSE
        DBMS_OUTPUT.PUT_LINE('Nu au fost plasate comenzi in acest an.');
    END IF;
    DBMS_OUTPUT.NEW_LINE;
END LOOP;

EXCEPTION
    WHEN exc_interval THEN
        RAISE_APPLICATION_ERROR(-20000, 'Interval invalid! Introduceti');
END analiza_interval;
/

```

Dbms Output Content:

```

Restaurantul a fost înființat în anul 2019. Analiza afișată va începe cu acest an.
Nu poate fi afișată analiza pe anii următori. Analiza se va termina cu anul curent.
-----
ANUL 2019
Suma incasari: 66

Nr de produse comandate din
- categoria bauturi -> 0 produse
- categoria desert -> 0 produse
- categoria frisura -> 0 produse
- categoria garnituri -> 0 produse
- categoria paste -> 3 produse
- categoria pizza -> 0 produse
- categoria salate -> 0 produse

Top 3 produse:
1. paste bolognese
2. -
3. -

Cel mai vandut produs: paste bolognese

Incasari din urma vanzarii produsului: 66
Procent din suma totala a incasarilor: 100%

Bucatarii care au gatit acest produs:
1. 200

-----
ANUL 2020
Suma incasari: 373

Nr de produse comandate din
- categoria bauturi -> 11 produse
- categoria desert -> 4 produse
- categoria frisura -> 0 produse
- categoria garnituri -> 0 produse

```

SQL Worksheet History projectSGBD

```

Worksheet Query Builder
produs1 := produs_top1(i);
IF produs1 != '0' THEN
    DBMS_OUTPUT.PUT_LINE('Cel mai vandut produs: ' || produs1); -
    DBMS_OUTPUT.NEW_LINE;
    analiza_produs(produs1, i); --apelam procedura definita anter
ELSE
    DBMS_OUTPUT.PUT_LINE('Nu au fost plasate comenzi in acest an.
END IF;
DBMS_OUTPUT.NEW_LINE;
END LOOP;

EXCEPTION
    WHEN exc_interval THEN
        RAISE_APPLICATION_ERROR(-20000, 'Interval invalid! Introducet
END analiza_interval;
END statistica;
/

```

```

BEGIN
    statistica.analiza_interval(2017, 2023);
    --statistica.analiza_interval(2017, 2018); --exceptie: interval invalid
END;
/

```

Script Output x Task completed in 0.072 seconds

Package STATISTICA compiled

Package Body STATISTICA compiled

PL/SQL procedure successfully completed.

Dbms Output x Buffer Size:20000 projectSGBD

Nr de produse comandate din

- categoria bauturi -> 11 produse
- categoria desert -> 4 produse
- categoria friptura -> 0 produse
- categoria garnituri -> 0 produse
- categoria paste -> 5 produse
- categoria pizza -> 2 produse
- categoria salate -> 6 produse

Top 3 produse:

1. apa
2. salata greceasca
3. paste carbonara

Cel mai vandut produs: apa

Incasari din urma vanzarii produsului: 35%

Procent din suma totala a incasarilor: 9.38%

Acest produs nu este gatit.

ANUL 2021

Suma incasari: 2683

Nr de produse comandate din

- categoria bauturi -> 69 produse
- categoria desert -> 15 produse
- categoria friptura -> 52 produse
- categoria garnituri -> 0 produse
- categoria paste -> 0 produse
- categoria pizza -> 16 produse
- categoria salate -> 7 produse

Top 3 produse:

1. friptura porc
2. suc

SQL Worksheet History projectSGBD

```

Worksheet Query Builder
produs1 := produs_top1(i);
IF produs1 != '0' THEN
    DBMS_OUTPUT.PUT_LINE('Cel mai vandut produs: ' || produs1); -
    DBMS_OUTPUT.NEW_LINE;
    analiza_produs(produs1, i); --apelam procedura definita anter
ELSE
    DBMS_OUTPUT.PUT_LINE('Nu au fost plasate comenzi in acest an.
END IF;
DBMS_OUTPUT.NEW_LINE;
END LOOP;

EXCEPTION
    WHEN exc_interval THEN
        RAISE_APPLICATION_ERROR(-20000, 'Interval invalid! Introducet
END analiza_interval;
END statistica;
/

```

```

BEGIN
    statistica.analiza_interval(2017, 2023);
    --statistica.analiza_interval(2017, 2018); --exceptie: interval invalid
END;
/

```

Script Output x Task completed in 0.072 seconds

Package STATISTICA compiled

Package Body STATISTICA compiled

PL/SQL procedure successfully completed.

Dbms Output x Buffer Size:20000 projectSGBD

Top 3 produse:

1. friptura porc
2. suc
3. vin

Cel mai vandut produs: friptura porc

Incasari din urma vanzarii produsului: 1176

Procent din suma totala a incasarilor: 43.83%

Bucatarii care au gatit acest produs:

1. 206
2. 204

ANUL 2022

Suma incasari: 0

Nr de produse comandate din

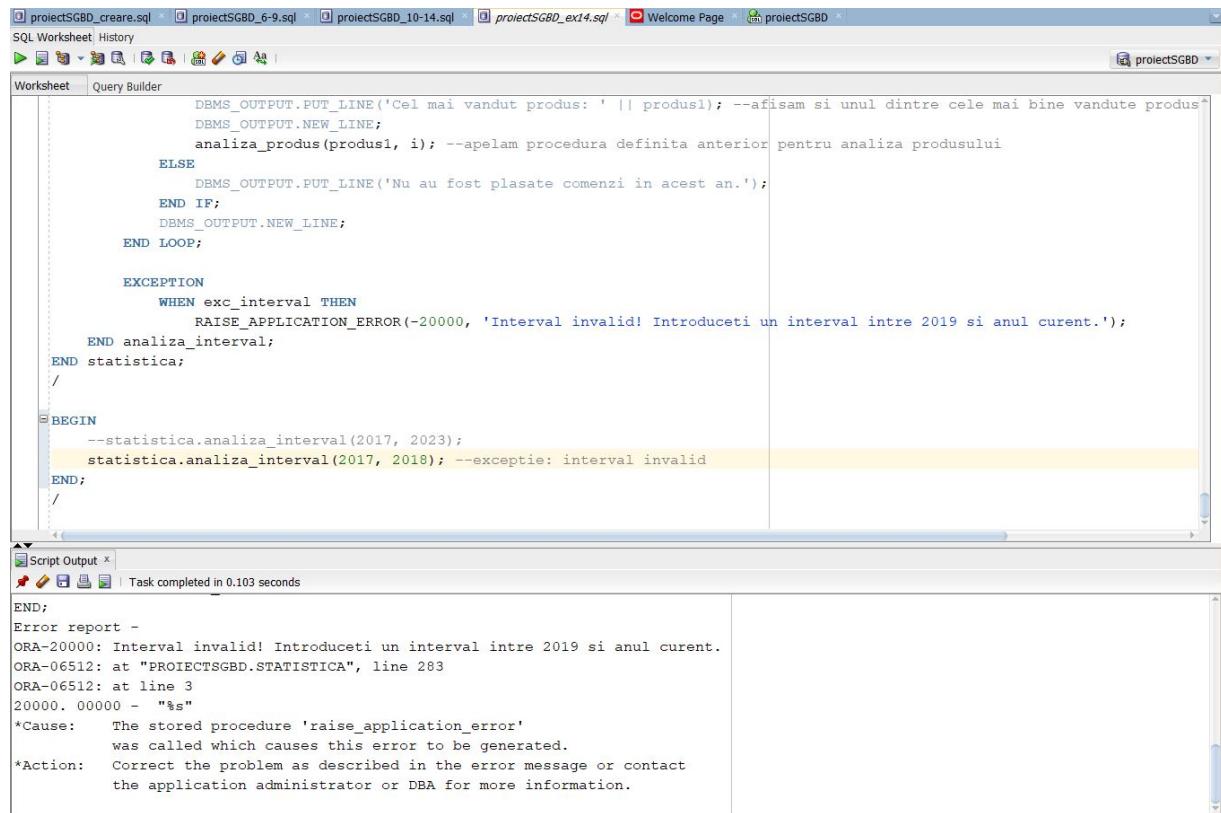
- categoria bauturi -> 0 produse
- categoria desert -> 0 produse
- categoria friptura -> 0 produse
- categoria garnituri -> 0 produse
- categoria paste -> 0 produse
- categoria pizza -> 0 produse
- categoria salate -> 0 produse

Top 3 produse:

1. -
2. -
3. -

Nu au fost plasate comenzi in acest an.

Exemplu apelare procedură analiza_interval din pachet cu un interval care nu este valid:



The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, several tabs are visible: projectSGBD_create.sql, projectSGBD_6-9.sql, projectSGBD_10-14.sql, projectSGBD_ex14.sql, Welcome Page, and projectSGBD. Below the tabs, the main area has two tabs: Worksheet and Query Builder. The Worksheet tab contains the following PL/SQL code:

```

DBMS_OUTPUT.PUT_LINE('Cel mai vandut produs: ' || produs1); --afisam si unul dintre cele mai bine vandute produse
DBMS_OUTPUT.NEW_LINE;
analiza_produs(produs1, i); --apelam procedura definita anterior pentru analiza produsului
ELSE
    DBMS_OUTPUT.PUT_LINE('Nu au fost plasate comenzi in acest an.');
END IF;
DBMS_OUTPUT.NEW_LINE;
END LOOP;

EXCEPTION
    WHEN exc_interval THEN
        RAISE_APPLICATION_ERROR(-20000, 'Interval invalid! Introduceti un interval intre 2019 si anul curent.');
END analiza_interval;
END statistica;
/

```

The code then begins a block labeled BEGIN:

```

BEGIN
    --statistica.analiza_interval(2017, 2023);
    statistica.analiza_interval(2017, 2018); --exceptie: interval invalid
END;
/

```

In the Script Output tab, the execution results are shown:

```

END;
Error report -
ORA-20000: Interval invalid! Introduceti un interval intre 2019 si anul curent.
ORA-06512: at "PROJECTSGBD.STATISTICA", line 283
ORA-06512: at line 3
20000. 00000 - "%s"
*Cause: The stored procedure 'raise_application_error'
    was called which causes this error to be generated.
*Action: Correct the problem as described in the error message or contact
    the application administrator or DBA for more information.

```