

PROIECT
SISTEME DE GESTIUNE A BAZELOR DE DATE
TEMA: GESTIUNEA UNUI LANȚ DE
RESTAURANTE

Lect. Univ. Dr. Gabriela Mihai
Student: Sora Andreea-Ioana
Seria 23, Grupa 234
Anul II, Semestrul I

CUPRINS

Cerința 1 – Prezentarea bazei de date.....	3
Cerința 2 – Diagrama entitate-relație.....	5
Cerința 3 – Diagrama conceptuală.....	6
Cerința 4 – Implementarea diagramei conceptuale în Oracle.....	6
Cerința 5 – Inserare date în tabele.....	14
Cerința 6 – Subprogram stocat - două tipuri de colecții.....	29
Cerința 7 – Subprogram stocat - un tip de cursor.....	33
Cerința 8 – Subprogram stocat de tip funcție - trei tabele.....	36
Cerința 9 – Subprogram stocat de tip procedură - cinci tabele.....	41
Cerința 10 – Trigger de tip LMD la nivel de comandă.....	48
Cerința 11 – Trigger de tip LMD la nivel de linie.....	53
Cerința 12 – Trigger de tip LDD.....	58
Cerința 13 – Pachet cu toate obiectele definite în cadrul proiectului.....	62
Cerința 14 – Pachet – 2 tipuri de date/2 funcții/2 proceduri.....	71
Bibliografie.....	87

1. Prezențați pe scurt baza de date (utilitatea ei).

În acest proiect vom analiza gestiunea unui lanț de restaurante. Aceasta va ajuta la o mai bună funcționare, la posibilitatea de a ține evidența comenzilor plasate de către fiecare client către fiecare restaurant și la gestionarea evenimentelor ce pot avea loc în fiecare dintre locațiile noastre.

Entitățile modelului sunt următoarele:

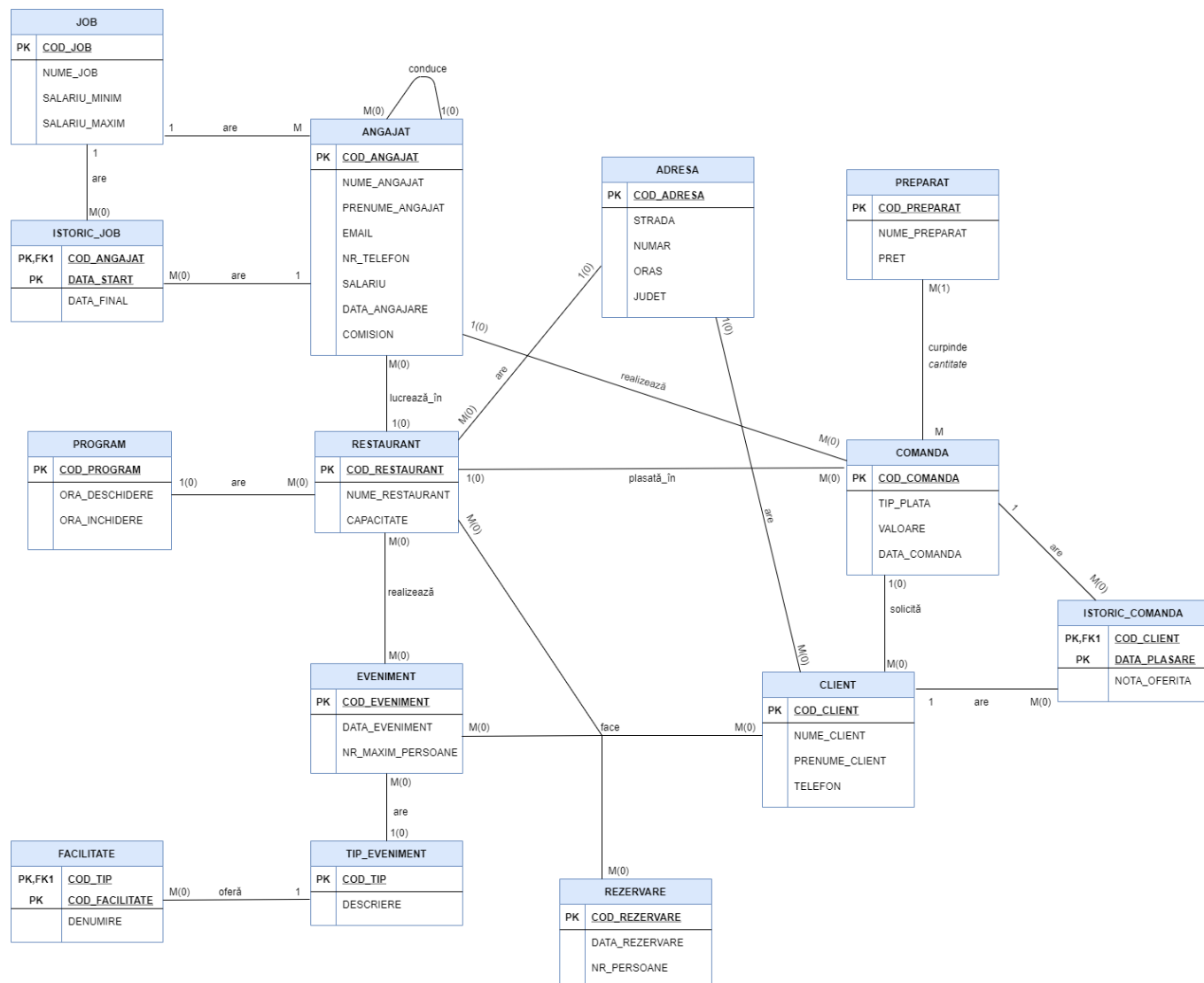
- RESTAURANT;
- EVENIMENT;
- TIP_EVENT;
- FACILITATE;
- PROGRAM;
- ADRESA;
- ANGAJAT;
- JOB;
- ISTORIC_JOB;
- CLIENT;
- REZERVARE;
- COMANDA;
- ISTORIC_COMANDA;
- PREPARAT.

Regulile de funcționare pentru acest model sunt următoarele:

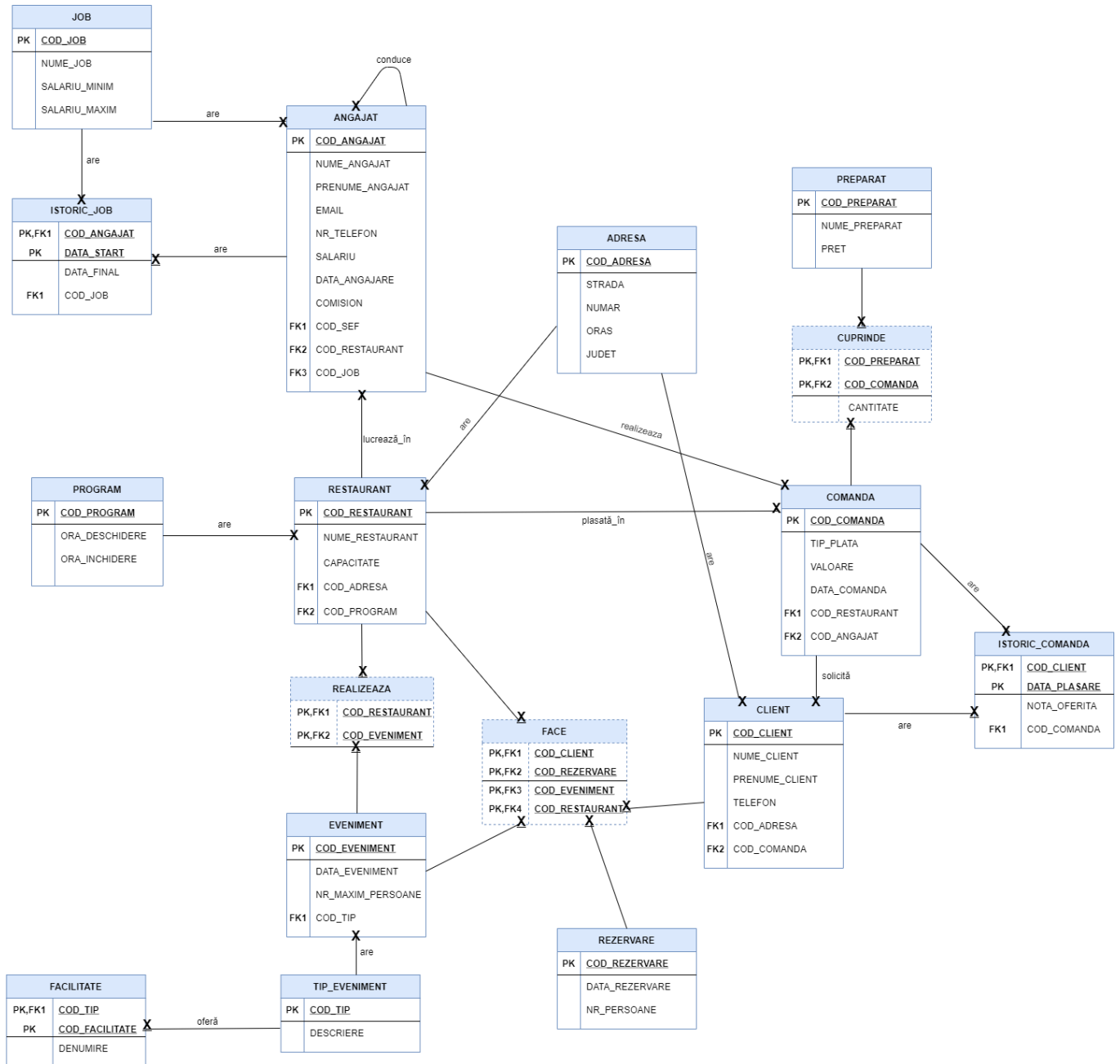
- În restaurante se pot organiza mai multe evenimente cărora le corespunde un tip de eveniment și, în funcție de tip, mai multe facilități;
- Pentru evenimentele care pot avea loc (nuntă, botez, majorat, petreceri de weekend, petreceri de 8 martie etc) se pot face mai multe rezervări. Vom reține inclusiv evenimentele pentru care nu s-a făcut nicio rezervare până la momentul actual;
- Un eveniment are o dată de programare, dar poate avea loc simultan în mai multe din locațiile noastre în cazul în care acest lucru este posibil (spre exemplu, evenimente de 8 martie);
- Angajații au un anumit job și un istoric unde vom ține evidența posturilor pe care aceștia le-au avut în restaurant. Data de angajare a fiecărui salariat va fi prima dată când acesta s-a angajat. În cazul în care cineva își schimbă job-ul, acest lucru se va consemna în istoric. Vom presupune că un salariat, o dată angajat într-unul din restaurante, va putea schimba job-ul doar în cadrul aceluiași restaurant. Pot exista angajați care nu au restaurantul setat;

- Un angajat nu poate să primească două job-uri diferite în aceeași zi. Salariul trebuie să fie cuprins între limitele salariale stabilite pentru fiecare job în parte;
- Nu pot exista angajați cu același email, același număr de telefon sau aceeași combinație nume-prenume (sunt unice);
- Restaurantele pot avea o adresă și un program de funcționare;
- Fiecare restaurant din lanțul nostru dispune de același meniu;
- Clienții pot face rezervări pentru evenimente și pot plasa comenzi, având de asemenea un istoric de comenzi. Un client nu poate solicita două comenzi diferite în aceeași zi. Toți clienții care solicită comenzi trebuie să aibă setată o adresă;
- La fiecare rezervare se va reține numărul de locuri pentru care clientul solicită acest lucru. Acesta poate face rezervare pentru nuntă/botez specificând numărul de invitați sau pentru evenimentele de alt tip/comune ce permit rezervări pentru un număr limitat de persoane, ținând cont de numărul maxim de persoane stabilit de fiecare restaurant;
- Comanda cuprinde diverse preparate, într-o anumită cantitate. Toate comenzile trebuie să cuprindă cel puțin un preparat. Comanda solicitată de fiecare client este realizată de un singur angajat, având job-ul de „bucătar”;
- Se vor reține o singură dată informațiile despre clienți alături de prima comanda (dacă acesta solicită o comanda înainte de a face o rezervare, altfel se vor reține doar datele de contact), urmând ca pe viitor, celelalte comenzi să se consemneze în istoricul comenzilor. Dacă un client există în baza de date și face prima oară o rezervare (și are adresa setată null), apoi o comandă, se va actualiza codul adresei sale. Pentru fiecare comandă, clienții oferă o nota/recenzie. Notele clienților vor fi numere naturale cuprinse între 1 și 10;
- Nu pot exista clienți cu același număr de telefon sau cu aceeași combinație nume-prenume (sunt unice);
- Inițial, valoarea comenzii va fi setată NULL, urmând ca mai apoi să fie calculată automat în funcție de prețul și cantitatea preparatelor pe care le cuprinde.

2. Realizați diagrama entitate-relație (ERD).



3. Pornind de la diagrama entitate-relație realizați **diagrama conceptuală** modelului propus, integrând toate attributele necesare.



4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, implementând toate constrângerile de integritate necesare (chei primare, cheile externe etc).

```
create table PROGRAM (
cod_program number(2),
```

```
ora_deschidere varchar2(6),
ora_inchidere varchar2(6),
constraint program_cod_pk primary key (cod_program),
constraint program_in_des_ck check(ora_inchidere>ora_deschidere)
);
```

```
create table ADRESA (
cod_adresa number(4),
strada varchar2(30) constraint null_strada not null,
numar number(3) constraint null_numar not null,
oras varchar2(30) constraint null_oras not null,
judet varchar2(30),
constraint adresa_cod_pk primary key (cod_adresa)
);
```

```
create table RESTAURANT (
cod_restaurant number(4),
nume_restaurant varchar2(30) constraint null_nume_res not null,
capacitate number(4),
cod_adresa number(4),
cod_program number(2),
constraint restaurant_cod_pk primary key (cod_restaurant),
constraint fk_res_adresa foreign key (cod_adresa) references ADRESA(cod_adresa),
constraint fk_res_program foreign key (cod_program) references
PROGRAM(cod_program)
);
```

```
create table TIP_EVENTIMENT (
cod_tip number(2),
descriere varchar2(30) constraint null_descriere not null,
constraint tip_pk primary key (cod_tip)
);
```

```
create table EVENTIMENT (
cod_eventiment number(4),
data_eventiment date,
nr_maxim_persoane number(4),
cod_tip number(2),
constraint eveniment_cod_pk primary key (cod_eventiment),
constraint fk_ev_tip foreign key (cod_tip) references TIP_EVENTIMENT(cod_tip)
);
```

```
create table REALIZEAZA (
```

```
cod_restaurant number(4),
cod_eveniment number(4),
constraint real_pk primary key (cod_restaurant, cod_eveniment),
constraint fk_real_res foreign key (cod_restaurant) references
RESTAURANT(cod_restaurant),
constraint fk_real_ev foreign key (cod_eveniment) references
EVENTIMENT(cod_eveniment)
);

create table FACILITATE (
cod_tip number(2),
cod_facilitate number(2),
denumire varchar2(50),
constraint facilitati_pk primary key (cod_tip, cod_facilitate),
constraint fk_fac_tip foreign key (cod_tip) references TIP_EVENTIMENT(cod_tip)
);

create table JOB (
cod_job number(4),
nume_job varchar2(35) constraint null_nume_job not null,
salariu_minim number(6),
salariu_maxim number(6),
constraint job_pk primary key (cod_job)
);

create table ANGAJAT (
cod_angajat number(4),
nume_angajat varchar2(25) constraint null_nume_ang not null,
prenume_angajat varchar2(25),
email varchar2(25) constraint null_email not null,
nr_telefon varchar2(15) constraint null_nr_tel not null,
salariu number(8,2),
data_angajare date default sysdate,
comision number(2,2),
cod_sef number(4),
cod_restaurant number(4),
cod_job number(4) constraint null_fk_job not null,
constraint ang_pk primary key (cod_angajat),
constraint fk_ang_ang foreign key (cod_sef) references ANGAJAT(cod_angajat),
constraint fk_ang_res foreign key (cod_restaurant) references
RESTAURANT(cod_restaurant),
constraint fk_ang_job foreign key (cod_job) references JOB(cod_job),
constraint unq_nume_prenume unique (nume_angajat,prenume_angajat),
```



```
constraint unq_email unique (email),  
constraint unq_nr_tel unique (nr_telefon),  
constraint ck_sal check (salariu>0)  
);
```

```
create table ISTORIC_JOB (  
cod_angajat number(4),  
data_start date default sysdate,  
data_final date,  
cod_job number(4) constraint null_job_istoric not null,  
constraint istoric_job_pk primary key (cod_angajat, data_start),  
constraint fk_istoric_job foreign key (cod_job) references JOB(cod_job),  
constraint fk_istoric_ang foreign key (cod_angajat) references ANGAJAT(cod_angajat)  
);
```

```
create table REZERVARE (  
cod_rezervare number(6),  
data_rezervare date default sysdate,  
nr_persoane number(3),  
constraint rez_pk primary key (cod_rezervare)  
);
```

```
create table COMANDA (  
cod_comanda number(4),  
tip_plata varchar2(20),  
valoare number(8,2),  
data_comanda date,  
cod_restaurant number(4),  
cod_angajat number(4),  
constraint pk_comanda primary key (cod_comanda),  
constraint fk_comanda_res foreign key (cod_restaurant) references  
RESTAURANT(cod_restaurant),  
constraint fk_comanda_ang foreign key (cod_angajat) references  
ANGAJAT(cod_angajat)  
);
```

```
create table CLIENT (  
cod_client number(4),  
nume_client varchar2(25) constraint null_nume_client not null,  
prenume_client varchar2(25),  
telefon varchar2(15) constraint null_telefon_client not null,  
cod_adresa number(4),  
cod_comanda number(4),
```

```
constraint client_pk primary key (cod_client),
constraint unq_num_prename_cl unique (nume_client,prename_client),
constraint fk_cl_adresa foreign key (cod_adresa) references ADRESA(cod_adresa),
constraint fk_cl_comanda foreign key (cod_comanda) references
COMANDA(cod_comanda),
constraint unq_tel unique (telefon)
);
```

```
create table FACE (
cod_client number(4),
cod_rezervare number(6),
cod_eveniment number(4),
cod_restaurant number(4),
constraint face_pk primary key (cod_client, cod_rezervare, cod_eveniment,
cod_restaurant),
constraint fk_face_client foreign key (cod_client) references CLIENT(cod_client),
constraint fk_face_rez foreign key (cod_rezervare) references
REZERVARE(cod_rezervare),
constraint fk_face_ev foreign key (cod_eveniment) references
EVENTIMENT(cod_eveniment),
constraint fk_face_res foreign key (cod_restaurant) references
RESTAURANT(cod_restaurant)
);
```

```
create table ISTORIC_COMANDA (
cod_client number(4),
data_plasare date default sysdate,
nota_oferita number(2),
cod_comanda number(4) constraint null_comanda_istoric not null,
constraint istoric_comanda_pk primary key (cod_client, data_plasare),
constraint fk_istoric_comanda foreign key (cod_comanda) references
COMANDA(cod_comanda),
constraint fk_istoric_cl foreign key (cod_client) references CLIENT(cod_client)
);
```

```
create table PREPARAT (
cod_preparat number(4),
nume_preparat varchar2(25) constraint null_preparat not null,
pret number(8,2) constraint null_pret not null,
constraint pk_preparat primary key (cod_preparat)
);
```

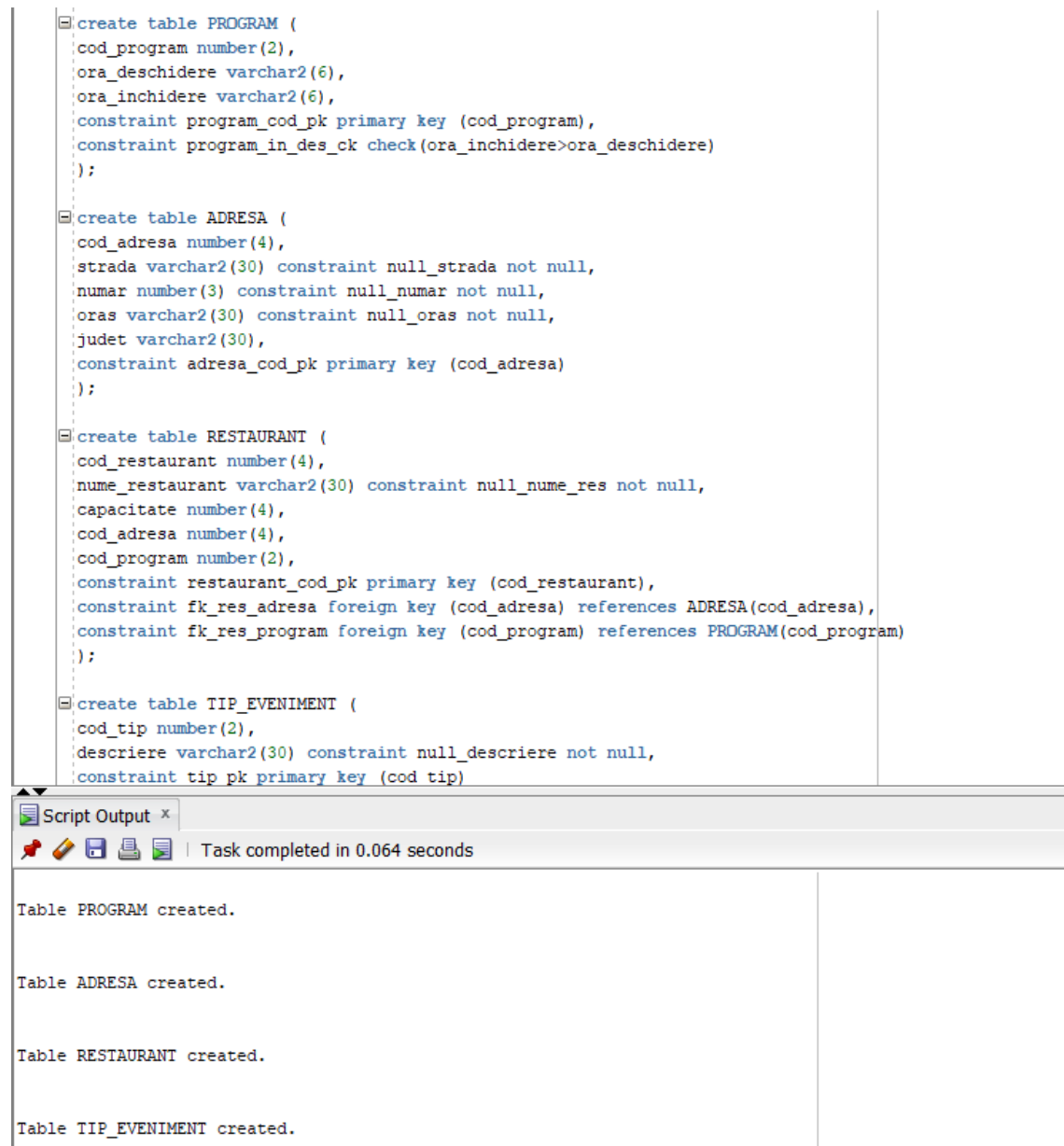
```
create table CUPRINDE (
```

```

cod_preparat number(4),
cod_comanda number(4),
cantitate number(2) constraint null_cantitate not null,
constraint pk_cuprinde primary key (cod_preparat, cod_comanda),
constraint fk_cup_prep foreign key (cod_preparat) references
PREPARAT(cod_preparat),
constraint fk_cup_com foreign key (cod_comanda) references
COMANDA(cod_comanda)
);

commit;

```



The screenshot displays a database IDE interface. The top pane shows a SQL script for creating four tables: PROGRAM, ADRESA, RESTAURANT, and TIP_EVENTIMENT. The script includes primary and foreign key constraints. The bottom pane, titled 'Script Output', shows the successful execution of the script, with messages indicating that each table was created. A status bar at the bottom of the output pane states 'Task completed in 0.064 seconds'.

```

create table PROGRAM (
cod_program number(2),
ora_deschidere varchar2(6),
ora_inchidere varchar2(6),
constraint program_cod_pk primary key (cod_program),
constraint program_in_des_ck check(ora_inchidere>ora_deschidere)
);

create table ADRESA (
cod_adresa number(4),
strada varchar2(30) constraint null_strada not null,
numar number(3) constraint null_numar not null,
oras varchar2(30) constraint null_oras not null,
judet varchar2(30),
constraint adresa_cod_pk primary key (cod_adresa)
);

create table RESTAURANT (
cod_restaurant number(4),
nume_restaurant varchar2(30) constraint null_nume_res not null,
capacitate number(4),
cod_adresa number(4),
cod_program number(2),
constraint restaurant_cod_pk primary key (cod_restaurant),
constraint fk_res_adresa foreign key (cod_adresa) references ADRESA(cod_adresa),
constraint fk_res_program foreign key (cod_program) references PROGRAM(cod_program)
);

create table TIP_EVENTIMENT (
cod_tip number(2),
descriere varchar2(30) constraint null_descriere not null,
constraint tip_pk primary key (cod_tip)
);

```

Script Output x

Task completed in 0.064 seconds

Table PROGRAM created.

Table ADRESA created.

Table RESTAURANT created.

Table TIP_EVENTIMENT created.

```

create table EVENIMENT (
  cod_eveniment number(4),
  nr_persoane number(4),
  cod_tip number(2),
  constraint eveniment_cod_pk primary key (cod_eveniment),
  constraint fk_ev_tip foreign key (cod_tip) references TIP_EVENTIMENT(cod_tip)
);

create table REALIZEAZA (
  cod_restaurant number(4),
  cod_eveniment number(4),
  constraint real_pk primary key (cod_restaurant, cod_eveniment),
  constraint fk_real_res foreign key (cod_restaurant) references RESTAURANT(cod_restaurant),
  constraint fk_real_ev foreign key (cod_eveniment) references EVENIMENT(cod_eveniment)
);

create table FACILITATE (
  cod_tip number(2),
  cod_facilitate number(2),
  denumire varchar2(50),
  constraint facilitati_pk primary key (cod_tip, cod_facilitate),
  constraint fk_fac_tip foreign key (cod_tip) references TIP_EVENTIMENT(cod_tip)
);

create table JOB (
  cod_job number(4),
  nume_job varchar2(35) constraint null_nume_job not null,
  salariu_minim number(6),
  salariu_maxim number(6),
  constraint job_pk primary key (cod_job)
);

```

Script Output x

Task completed in 0.073 seconds

Table EVENIMENT created.

Table REALIZEAZA created.

Table FACILITATE created.

Table JOB created.

```

create table ANGAJAT (
  cod_angajat number(4),
  nume_angajat varchar2(25) constraint null_nume_ang not null,
  prenume_angajat varchar2(25),
  email varchar2(25) constraint null_email not null,
  nr_telefon varchar2(15) constraint null_nr_tel not null,
  salariu number(8,2),
  data_angajare date default sysdate,
  comision number(2,2),
  cod_sef number(4),
  cod_restaurant number(4),
  cod_job number(4) constraint null_fk_job not null,
  constraint ang_pk primary key (cod_angajat),
  constraint fk_ang_ang foreign key (cod_sef) references ANGAJAT(cod_angajat),
  constraint fk_ang_res foreign key (cod_restaurant) references RESTAURANT(cod_restaurant),
  constraint fk_ang_job foreign key (cod_job) references JOB(cod_job),
  constraint unq_nume_prenume unique (nume_angajat, prenume_angajat),
  constraint unq_email unique (email),
  constraint unq_nr_tel unique (nr_telefon),
  constraint ck_sal check (salariu > 0)
);

create table ISTORIC_JOB (
  cod_angajat number(4),
  data_start date default sysdate,
  data_final date,
  cod_job number(4) constraint null_job_istoric not null,
  constraint istoric_job_pk primary key (cod_angajat, data_start),
  constraint fk_istoric_job foreign key (cod_job) references JOB(cod_job),
  constraint fk_istoric_ang foreign key (cod_angajat) references ANGAJAT(cod_angajat)
);

```

Script Output x

Task completed in 0.07 seconds

Table FACILITATE created.

Table JOB created.

Table ANGAJAT created.

Table ISTORIC_JOB created.

<pre> create table REZERVARE (cod_rezervare number(6), data_rezervare date default sysdate, data_eveniment date, constraint rez_pk primary key (cod_rezervare), constraint ck_date_rez check (data_eveniment>data_rezervare)); create table COMANDA (cod_comanda number(4), tip_plata varchar2(20), valoare number(8,2), data_comanda date, cod_restaurant number(4), cod_angajat number(4), constraint pk_comanda primary key (cod_comanda), constraint fk_comanda_res foreign key (cod_restaurant) references RESTAURANT(cod_restaurant), constraint fk_comanda_ang foreign key (cod_angajat) references ANGAJAT(cod_angajat)); create table CLIENT (cod_client number(4), nume_client varchar2(25) constraint null_nume_client not null, prenume_client varchar2(25), telefon varchar2(15) constraint null_telefon_client not null, cod_adresa number(4), cod_comanda number(4), constraint client_pk primary key (cod_client), constraint unq_nume_prenume_cl unique (nume_client,prenume_client), constraint fk_cl_adresa foreign key (cod_adresa) references ADRESA(cod_adresa), constraint fk_cl_comanda foreign key (cod_comanda) references COMANDA(cod_comanda), constraint unq_tel unique (telefon)); </pre>	
<p>Script Output x</p> <p>Task completed in 0.077 seconds</p> <p>Table REZERVARE created.</p> <p>Table COMANDA created.</p> <p>Table COMANDA dropped.</p> <p>Table COMANDA created.</p> <p>Table CLIENT created.</p>	
<pre>); create table FACE (cod_client number(4), cod_rezervare number(6), cod_eveniment number(4), cod_restaurant number(4), constraint face_pk primary key (cod_client, cod_rezervare, cod_eveniment, cod_restaurant), constraint fk_face_client foreign key (cod_client) references CLIENT(cod_client), constraint fk_face_rez foreign key (cod_rezervare) references REZERVARE(cod_rezervare), constraint fk_face_ev foreign key (cod_eveniment) references EVENIMENT(cod_eveniment), constraint fk_face_res foreign key (cod_restaurant) references RESTAURANT(cod_restaurant)); create table ISTORIC_COMANDA (cod_client number(4), data_plasare date default sysdate, nota_oferita number(2), cod_comanda number(4) constraint null_comanda_istoric not null, constraint istoric_comanda_pk primary key (cod_client, data_plasare), constraint fk_istoric_comanda foreign key (cod_comanda) references COMANDA(cod_comanda), constraint fk_istoric_cl foreign key (cod_client) references CLIENT(cod_client)); create table PREPARAT (cod_preparat number(4), nume_preparat varchar2(25) constraint null_preparat not null, pret number(8,2) constraint null_pret not null, constraint pk_preparat primary key (cod_preparat)); create table CUPRINDE (cod_preparat number(4), cod_comanda number(4), </pre>	
<p>Script Output x</p> <p>Task completed in 0.073 seconds</p> <p>Table FACE created.</p> <p>Table ISTORIC_COMANDA created.</p> <p>Table PREPARAT created.</p> <p>Table CUPRINDE created.</p>	

5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).

```
insert into PROGRAM
values (1,'08:00','22:00');
```

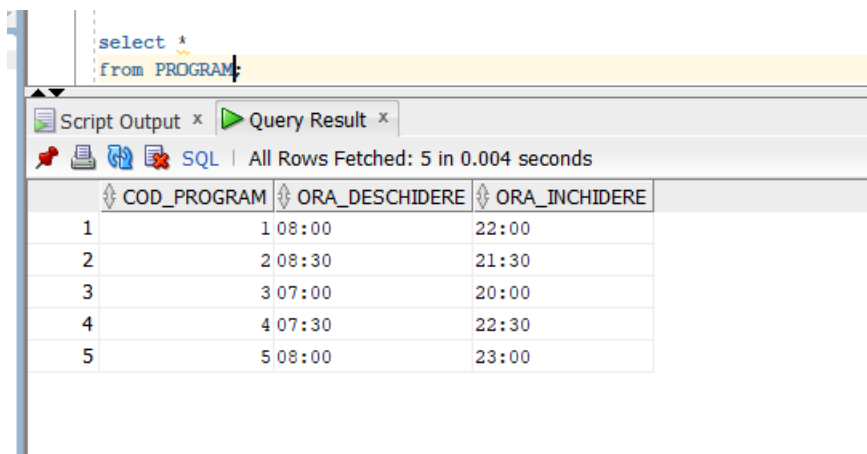
```
insert into PROGRAM
values (2,'08:30','21:30');
```

```
insert into PROGRAM
values (3,'07:00','20:00');
```

```
insert into PROGRAM
values (4,'07:30','22:30');
```

```
insert into PROGRAM
values (5,'08:00','23:00');
```

```
commit;
```



The screenshot shows a SQL query result in a database client. The query is `select * from PROGRAM`. The result set contains 5 rows of data. The columns are `COD_PROGRAM`, `ORA_DESCHIDERE`, and `ORA_INCHIDERE`. The data is as follows:

	COD_PROGRAM	ORA_DESCHIDERE	ORA_INCHIDERE
1	1	08:00	22:00
2	2	08:30	21:30
3	3	07:00	20:00
4	4	07:30	22:30
5	5	08:00	23:00

```
insert into ADRESA
values (8,'Str. Mihai Eminescu',23,'Ploiesti','Prahova');
```

```
insert into ADRESA
values (9,'Str. Cuza Voda',5,'Campina',null);
```

```
insert into ADRESA
values (10,'Str. Nicolae Balcescu',250,'Sinaia','Prahova');
```

```
insert into ADRESA
values (11,'Str. 1 Decembrie',109,'Ploiesti','Prahova');
```

```
insert into ADRESA
values (12,'Str. Ion Creanga',7,'Busteni',null);
```

```
insert into ADRESA
values (13,'Str. Ion Luca Caragiale',108,'Ploiesti','Prahova');
```

```
insert into ADRESA
values (14,'Str. 25 Decembrie',148,'Ploiesti','Prahova');
```

```
insert into ADRESA
values (15,'Str. Mihai Bravu',89,'Ploiesti','Prahova');
```

```
insert into ADRESA
values (16,'Str. Matei Basarab',657,'Ploiesti',null);
```

insert into ADRESA

values (17,'Str. Nicolae Balcescu',128,'Ploiesti','Prahova');

insert into ADRESA

values (18,'Str. Carol I',12,'Ploiesti',null);

insert into ADRESA

values (19,'Str. Unirii',349,'Ploiesti','Prahova');

commit;

select *
from ADRESA;

Script Output x Query Result x

SQL | All Rows Fetched: 12 in 0.004 seconds

	COD_ADRESA	STRADA	NUMAR	ORAS	JUDET
1	8	Str. Mihai Eminescu	23	Ploiesti	Prahova
2	9	Str. Cuza Voda	5	Campina	(null)
3	10	Str. Nicolae Balcescu	250	Sinaia	Prahova
4	11	Str. 1 Decembrie	109	Ploiesti	Prahova
5	12	Str. Ion Creanga	7	Busteni	(null)
6	13	Str. Ion Luca Caragiale	108	Ploiesti	Prahova
7	14	Str. 25 Decembrie	148	Ploiesti	Prahova
8	15	Str. Mihai Bravu	89	Ploiesti	Prahova
9	16	Str. Matei Basarab	657	Ploiesti	(null)
10	17	Str. Nicolae Balcescu	128	Ploiesti	Prahova
11	18	Str. Carol I	12	Ploiesti	(null)
12	19	Str. Unirii	349	Ploiesti	Prahova

insert into RESTAURANT

values (101,'Da Vinci',400,12,1);

insert into RESTAURANT

values (102,'Best',500,8,5);

insert into RESTAURANT

values (103,'Mamaliguta',350,9,3);

insert into RESTAURANT

values (104,'Lazarini',600,11,1);

insert into RESTAURANT

values (105,'Trattoria',600,10,null);

insert into RESTAURANT

values (106,'Toscana',450,null,2);

commit;

select *
from restaurant;

Script Output x Query Result x

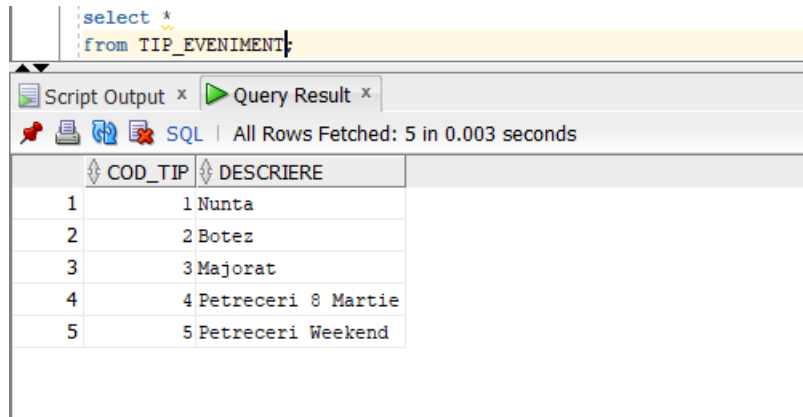
SQL | All Rows Fetched: 6 in 0.003 seconds

	COD_RESTAURANT	NUME_RESTAURANT	CAPACITATE	COD_ADRESA	COD_PROGRAM
1	101	Da Vinci	400	12	1
2	102	Best	500	8	5
3	103	Mamaliguta	350	9	3
4	104	Lazarini	600	11	1
5	105	Trattoria	600	10	(null)
6	106	Toscana	450	(null)	2

```
insert into TIP_EVENTIMENT
values (1,'Nunta');
```

```
insert into TIP_EVENTIMENT
values (2,'Botez');
```

```
insert into TIP_EVENTIMENT
values (3,'Majorat');
```



The screenshot shows a SQL query result in a database management tool. The query is `select * from TIP_EVENTIMENT;`. The result is displayed in a table with two columns: `COD_TIP` and `DESCRIERE`. There are 5 rows of data.

COD_TIP	DESCRIERE
1	1 Nunta
2	2 Botez
3	3 Majorat
4	4 Petreceri 8 Martie
5	5 Petreceri Weekend

```
insert into TIP_EVENTIMENT
values (4,'Petreceri 8 Martie');
```

```
insert into TIP_EVENTIMENT
values (5,'Petreceri Weekend');
```

```
commit;
```

```
insert into EVENTIMENT
values (300,to_date('02-10-2021','dd-mm-yyyy'),100,1);
```

```
insert into EVENTIMENT
values (301,to_date('12-05-2022','dd-mm-yyyy'),200,1);
```

```
insert into EVENTIMENT
values (302,to_date('17-07-2021','dd-mm-yyyy'),70,5);
```

```
insert into EVENTIMENT
values (303,to_date('08-03-2022','dd-mm-yyyy'),200,4);
```

```
insert into EVENTIMENT
values (304,to_date('23-02-2022','dd-mm-yyyy'),null,3);
```

```
insert into EVENTIMENT
values (305,to_date('30-04-2022','dd-mm-yyyy'),300,2);
```

```
insert into EVENTIMENT
values (306,to_date('08-06-2022','dd-mm-yyyy'),200,2);
```

```
insert into EVENTIMENT
values (307,to_date('24-07-2021','dd-mm-yyyy'),70,5);
```

```
insert into EVENTIMENT
values (308,to_date('01-08-2021','dd-mm-yyyy'),70,5);
```

```
commit;
```


select *
from EVENIMENT;

Script Output x Query Result x

SQL | All Rows Fetched: 9 in 0.003 seconds

	COD_EVENTIMENT	DATA_EVENTIMENT	NR_MAXIM_PERSOANE	COD_TIP
1	300	02-OCT-21	100	1
2	301	12-MAY-22	200	1
3	302	17-JUL-21	70	5
4	303	08-MAR-22	200	4
5	304	23-FEB-22	(null)	3
6	305	30-APR-22	300	2
7	306	08-JUN-22	200	2
8	308	01-AUG-21	70	5
9	307	24-JUL-21	70	5

insert into REALIZEAZA
values (101,307);

insert into REALIZEAZA
values (102,307);

insert into REALIZEAZA
values (104,303);

insert into REALIZEAZA
values (106,303);

insert into REALIZEAZA
values (102,303);

insert into REALIZEAZA
values (105,302);

insert into REALIZEAZA
values (104,304);

insert into REALIZEAZA
values (102,300);

insert into REALIZEAZA
values (103,301);

insert into REALIZEAZA
values (103,306);

insert into REALIZEAZA
values (101,305);

insert into REALIZEAZA
values (106,302);

insert into REALIZEAZA
values (101,308);

commit;

select *
from REALIZEAZA;

Script Output x Query Result x

SQL | All Rows Fetched: 13 in 0.002 seconds

	COD_RESTAURANT	COD_EVENTIMENT
1	101	305
2	101	307
3	101	308
4	102	300
5	102	303
6	102	307
7	103	301
8	103	306
9	104	303
10	104	304
11	105	302
12	106	302
13	106	303

```
insert into FACILITATE
values (1,1,'aranjament floral');
```

```
insert into FACILITATE
values (4,2,'candy bar');
```

```
insert into FACILITATE
values (5,3,'o cafea gratis/invitat');
```

```
insert into FACILITATE
values (3,4,'baloane cu heliu');
```

```
insert into FACILITATE
values (3,5,'o sticla de apa/masa');
```

```
insert into FACILITATE
values (3,6,null);
```

```
commit;
```

The screenshot shows a SQL query result for the FACILITATE table. The query is 'select * from FACILITATE;'. The result is displayed in a table with columns: COD_TIP, COD_FACILITATE, and DENUMIRE. The data is as follows:

	COD_TIP	COD_FACILITATE	DENUMIRE
1	1	1	aranjament floral
2	4	2	candy bar
3	3	4	baloane cu heliu
4	5	3	o cafea gratis/invitat
5	3	5	o sticla de apa/masa
6	3	6	(null)

```
insert into JOB
values (1000,'manager',10000,30000);
```

```
insert into JOB
values (1001,'bucatar',2500,15000);
```

```
insert into JOB
values (1002,'casier',1500,4000);
```

```
insert into JOB
values (1003,'personal curatenie',1000,2000);
```

```
insert into JOB
values (1004,'ospatar',1500,4000);
```

```
insert into JOB
values (1005,'sef de sala',null,10400);
```

```
insert into JOB
values (1006,'ajutor bucatar',2000,9000);
```

```
commit;
```

The screenshot shows a SQL query result for the JOB table. The query is 'select * from job;'. The result is displayed in a table with columns: COD_JOB, NUME_JOB, SALARIU_MINIM, and SALARIU_MAXIM. The data is as follows:

	COD_JOB	NUME_JOB	SALARIU_MINIM	SALARIU_MAXIM
1	1000	manager	10000	30000
2	1001	bucatar	2500	15000
3	1002	casier	1500	4000
4	1003	personal curatenie	1000	2000
5	1004	ospatar	1500	4000
6	1005	sef de sala	(null)	10400
7	1006	ajutor bucatar	2000	9000

```
insert into ANGAJAT
```

```
values (10,'Popescu','Ion','email1@yahoo.com','0728647839',25000,to_date('02-10-2020','dd-mm-yyyy'),null,null,101,1000);
```

```
insert into ANGAJAT
```

```
values (11,'Gheorghe','Mihai','email2@yahoo.com','0728622839',10400,to_date('10-12-2020','dd-mm-yyyy'),0.1,10,101,1005);
```

```
insert into ANGAJAT
```

```
values (12,'Ionescu','Ana','email3@yahoo.com','0738689839',3000,to_date('02-01-2021','dd-mm-yyyy'),null,11,101,1002);
```

```
insert into ANGAJAT
```

```
values (13,'Neagu','Marian','email4@yahoo.com','0728647009',2500,to_date('12-03-2021','dd-mm-yyyy'),0.1,11,101,1004);
```

```
insert into ANGAJAT
```

```
values (14,'Toma','Daniela','email5@yahoo.com','0728678901',1900,to_date('02-09-2020','dd-mm-yyyy'),null,11,101,1003);
```

```
insert into ANGAJAT
```

```
values (15,'Petrescu','Marius','email6@yahoo.com','0712345678',9000,to_date('12-01-2021','dd-mm-yyyy'),0.35,11,101,1001);
```

```
insert into ANGAJAT
```

```
values (16,'Lazar','Nicoleta','email7@yahoo.com','0711223344',9000,to_date('01-02-2021','dd-mm-yyyy'),0.2,11,101,1001);
```

```
insert into ANGAJAT
```

```
values (17,'Ion','Dan','email8@yahoo.com','0701234985',5000,to_date('18-04-2021','dd-mm-yyyy'),null,15,101,1006);
```

```
insert into ANGAJAT
```

```
values (18,'Pop','Vlad','email9@yahoo.com','0789634185',20000,to_date('29-10-2020','dd-mm-yyyy'),null,null,104,1000);
```

```
insert into ANGAJAT
```

```
values (19,'Trandafir','Corina','email10@yahoo.com','0798765432',10300,to_date('02-01-2021','dd-mm-yyyy'),0.12,18,104,1005);
```

```
insert into ANGAJAT
```

```
values (20,'Dragos','Cristina','email11@yahoo.com','0798765431',14800,to_date('09-01-2021','dd-mm-yyyy'),null,19,104,1001);
```

```
insert into ANGAJAT
```

```
values (21,'Sorescu','Flavius','email12@yahoo.com','0798765435',4600,to_date('22-01-2021','dd-mm-yyyy'),null,20,104,1006);
```

```
insert into ANGAJAT
```

```
values (22,'Popescu','Ioana','email13@yahoo.com','0798765439',3300,to_date('10-06-2020','dd-mm-yyyy'),0.11,19,104,1002);
```

```
insert into ANGAJAT
```

```
values (23,'Ifrim','George','email14@yahoo.com','0789634188',20300,to_date('29-11-2020','dd-mm-yyyy'),0.3,null,102,1000);
```

```
insert into ANGAJAT
```

```
values (24,'Vasile','Tudor','email15@yahoo.com','0798765430',9080,to_date('02-03-2021','dd-mm-yyyy'),null,23,102,1005);
```

```
insert into ANGAJAT
```

```
values (25,'Dragos','Gheorghe','email16@yahoo.com','0798765400',8000,to_date('24-01-2021','dd-mm-yyyy'),null,24,102,1001);
```

```
insert into ANGAJAT
```

```
values (26,'Sora','George','email17@yahoo.com','0789634100',20900,to_date('02-11-2020','dd-mm-yyyy'),0.1,null,106,1000);
```

```
insert into ANGAJAT
```

```
values (27,'Podariu','Ionel','email18@yahoo.com','0798765411',9100,to_date('24-02-2021','dd-mm-yyyy'),null,26,106,1001);
```

```
insert into ANGAJAT
```

```
values (28,'Sandu','Tiberiu','email20@yahoo.com','0789000000',7900,to_date('06-09-2020','dd-mm-yyyy'),null,26,106,1001);
```

```
insert into ANGAJAT
```

```
values (29,'Dragomirescu','Laur','email21@yahoo.com','0781111111',21900,to_date('03-10-2020','dd-mm-yyyy'),0.1,null,105,1000);
```

```
insert into ANGAJAT
```

```
values (30,'Ion','Mihail','email22@yahoo.com','0781111112',9900,to_date('23-02-2021','dd-mm-yyyy'),null,29,105,1001);
```

```
insert into ANGAJAT
```

```
values (31,'Petrescu','Laurentiu','email23@yahoo.com','0781111113',18000,to_date('13-12-2020','dd-mm-yyyy'),0.12,null,103,1000);
```

```
insert into ANGAJAT
```

```
(cod_angajat,nume_angajat,prenume_angajat,email,nr_telefon,salariu,comision,cod_sef,cod_restaurant,cod_job)
```

```
values (32,'Branescu','Ionut','email24@yahoo.com','0781111117',8700,null,31,103,1001);
```

```
--data_angajare este by default data curenta
```

```
insert into ANGAJAT
```

```
values (33,'Bojici','Matei','email25@yahoo.com','0781111129',18000,to_date('13-10-2020','dd-mm-yyyy'),null,null,null,1000);
```

```
commit;
```

```
select *
from angajat;
```

Script Output x Query Result x

SQL All Rows Fetched: 24 in 0.003 seconds

	COD_ANGAJAT	NUME_ANGAJAT	PRENUME_ANGAJAT	EMAIL	NR_TELEFON	SALARIU	DATA_ANGAJARE	COMISION	COD_SEF	COD_RESTAURANT	COD_JOB
1	27	Podariu	Ionel	email118@yahoo.com	0798765411	9100	24-FEB-21	(null)	26	106	1001
2	10	Popescu	Ion	email11@yahoo.com	0728647839	25000	02-OCT-20	(null)	(null)	101	1000
3	11	Gheorghe	Mihai	email12@yahoo.com	0728622839	10400	10-DEC-20	0.1	10	101	1005
4	12	Ionescu	Ana	email13@yahoo.com	0738689839	3000	02-JAN-21	(null)	11	101	1002
5	13	Neagu	Marian	email14@yahoo.com	0728647009	2500	12-MAR-21	0.1	11	101	1004
6	14	Toma	Daniela	email15@yahoo.com	0728678901	1995	02-SEP-20	(null)	11	101	1003
7	15	Petrescu	Marius	email16@yahoo.com	0712345678	9000	12-JAN-21	0.35	11	101	1001
8	16	Lazar	Nicoleta	email17@yahoo.com	0711223344	9000	01-FEB-21	0.2	11	101	1001
9	17	Ion	Dan	email18@yahoo.com	0701234565	5000	18-APR-21	(null)	15	101	1006
10	18	Pop	Vlad	email19@yahoo.com	0789634185	21000	29-OCT-20	(null)	(null)	104	1000
11	19	Trandafir	Corina	email110@yahoo.com	0798765432	10300	02-JAN-21	0.12	18	104	1005
12	20	Dragos	Cristina	email111@yahoo.com	0798765431	14800	09-JAN-21	(null)	19	104	1001
13	21	Sorescu	Flavius	email112@yahoo.com	0798765435	4600	22-JAN-21	(null)	20	104	1006
14	22	Popescu	Ioana	email113@yahoo.com	0798765439	3465	10-JUN-20	0.11	19	104	1002
15	23	Ifrim	George	email114@yahoo.com	0789634188	20300	29-NOV-20	0.3	(null)	102	1000
16	24	Vasile	Tudor	email115@yahoo.com	0798765430	9080	02-MAR-21	(null)	23	102	1005
17	25	Dragos	Gheorghe	email116@yahoo.com	0798765400	8000	24-JAN-21	(null)	24	102	1001
18	26	Sora	George	email117@yahoo.com	0789634100	20900	02-NOV-20	0.1	(null)	106	1000
19	33	Bojici	Matei	email125@yahoo.com	0781111129	18000	13-OCT-20	(null)	(null)	(null)	1000
20	28	Sandu	Tiberiu	email120@yahoo.com	0789000000	7900	06-SEP-20	(null)	26	106	1001
21	29	Dragomirescu	Laur	email121@yahoo.com	0781111111	22995	03-OCT-20	0.1	(null)	105	1000
22	30	Ion	Mihail	email122@yahoo.com	0781111112	9900	23-FEB-21	(null)	29	105	1001
23	31	Petrescu	Laurentiu	email123@yahoo.com	0781111113	18900	13-DEC-20	0.12	(null)	103	1000
24	32	Branescu	Ionut	email124@yahoo.com	0781111117	8700	18-MAY-21	(null)	31	103	1001

```
insert into ISTORIC_JOB
```

```
values (23,to_date('29-11-2020','dd-mm-yyyy'),to_date('10-02-2021','dd-mm-yyyy'),1005);
```

```
insert into ISTORIC_JOB
```

```
values (23,to_date('11-02-2021','dd-mm-yyyy'),null,1000);
```

```
insert into ISTORIC_JOB
```

```
values (16,to_date('01-02-2021','dd-mm-yyyy'),to_date('29-03-2021','dd-mm-yyyy'),1006);
```

```
insert into ISTORIC_JOB
```

```
values (16,to_date('30-03-2021','dd-mm-yyyy'),null,1001);
```

```
insert into ISTORIC_JOB
```

```
values (26,to_date('02-11-2020','dd-mm-yyyy'),to_date('10-02-2021','dd-mm-yyyy'),1005);
```

```
insert into ISTORIC_JOB
```

```
values (26,to_date('11-02-2021','dd-mm-yyyy'),null,1000);
```

```
insert into ISTORIC_JOB
```

```
values (10,to_date('02-10-2020','dd-mm-yyyy'),null,1000);
```

```
insert into ISTORIC_JOB
```

```
values (11,to_date('10-12-2020','dd-mm-yyyy'),null,1005);
```

```
insert into ISTORIC_JOB
```

```
values (12,to_date('02-01-2021','dd-mm-yyyy'),null,1002);
```

```
insert into ISTORIC_JOB
```

```
values (13,to_date('12-03-2021','dd-mm-yyyy'),null,1004);
```

```
insert into ISTORIC_JOB
```

```
values (14,to_date('02-09-2020','dd-mm-yyyy'),null,1003);
```

```
insert into ISTORIC_JOB
```

```
values (15,to_date('12-01-2021','dd-mm-yyyy'),null,1001);
```

```
insert into ISTORIC_JOB
values (17,to_date('18-04-2021','dd-mm-yyyy'),null,1006);

insert into ISTORIC_JOB
values (18,to_date('29-10-2020','dd-mm-yyyy'),null,1000);

insert into ISTORIC_JOB
values (19,to_date('02-01-2021','dd-mm-yyyy'),null,1005);

insert into ISTORIC_JOB
values (20,to_date('09-01-2021','dd-mm-yyyy'),null,1001);

insert into ISTORIC_JOB
values (21,to_date('22-01-2021','dd-mm-yyyy'),null,1006);

insert into ISTORIC_JOB
values (22,to_date('10-06-2020','dd-mm-yyyy'),null,1002);

insert into ISTORIC_JOB
values (24,to_date('02-03-2021','dd-mm-yyyy'),null,1005);

insert into ISTORIC_JOB
values (25,to_date('24-01-2021','dd-mm-yyyy'),null,1001);

insert into ISTORIC_JOB
values (27,to_date('24-02-2021','dd-mm-yyyy'),null,1001);

insert into ISTORIC_JOB
values (28,to_date('06-09-2020','dd-mm-yyyy'),null,1001);

insert into ISTORIC_JOB
values (29,to_date('03-10-2020','dd-mm-yyyy'),null,1000);

insert into ISTORIC_JOB
values (30,to_date('23-02-2021','dd-mm-yyyy'),null,1001);

insert into ISTORIC_JOB
values (31,to_date('13-12-2020','dd-mm-yyyy'),null,1000);

insert into ISTORIC_JOB
values (32,to_date('18-05-2021','dd-mm-yyyy'),null,1001);

commit;
```

```
select *
from istoric_job;
```

Script Output x Query Result x

SQL | All Rows Fetched: 26 in 0.003 seconds

	COD_ANGAJAT	DATA_START	DATA_FINAL	COD_JOB
1	10	02-OCT-20	(null)	1000
2	11	10-DEC-20	(null)	1005
3	12	02-JAN-21	(null)	1002
4	13	12-MAR-21	(null)	1004
5	14	02-SEP-20	(null)	1003
6	15	12-JAN-21	(null)	1001
7	17	18-APR-21	(null)	1006
8	18	29-OCT-20	(null)	1000
9	19	02-JAN-21	(null)	1005
10	20	09-JAN-21	(null)	1001
11	21	22-JAN-21	(null)	1006
12	22	10-JUN-20	(null)	1002
13	24	02-MAR-21	(null)	1005
14	25	24-JAN-21	(null)	1001
15	27	24-FEB-21	(null)	1001
16	28	06-SEP-20	(null)	1001
17	29	03-OCT-20	(null)	1000
18	30	23-FEB-21	(null)	1001
19	31	13-DEC-20	(null)	1000
20	32	18-MAY-21	(null)	1001
21	23	29-NOV-20	10-FEB-21	1005
22	23	11-FEB-21	(null)	1000
23	16	01-FEB-21	29-MAR-21	1006
24	16	30-MAR-21	(null)	1001
25	26	02-NOV-20	10-FEB-21	1005
26	26	11-FEB-21	(null)	1000

```
insert into REZERVARE
values (700,to_date('02-04-2021','dd-mm-yyyy'),2);
```

```
insert into REZERVARE
values (701,to_date('02-05-2021','dd-mm-yyyy'),1);
```

```
insert into REZERVARE
values (702,to_date('05-01-2021','dd-mm-yyyy'),2);
```

```
insert into REZERVARE
values (703,to_date('23-02-2021','dd-mm-yyyy'),4);
```

```
insert into REZERVARE (cod_rezervare,nr_persoane)
values (704,2);
```

--data_rezervare este by default data curenta

```
insert into REZERVARE
values (705,to_date('14-03-2021','dd-mm-yyyy'),2);
```

```
insert into REZERVARE
values (706,to_date('25-01-2021','dd-mm-yyyy'),3);
```

```
insert into REZERVARE
values (707,to_date('13-01-2021','dd-mm-yyyy'),150);
```

```

insert into REZERVARE
values (708,to_date('21-01-2021','dd-mm-yyyy'),200);

insert into REZERVARE
values (709,to_date('10-02-2021','dd-mm-yyyy'),90);

insert into REZERVARE
values (710,to_date('16-01-2021','dd-mm-yyyy'),90);

insert into REZERVARE
values (711,to_date('28-01-2021','dd-mm-yyyy'),95);

insert into REZERVARE
values (712,to_date('28-04-2021','dd-mm-yyyy'),5);

insert into REZERVARE
values (713,to_date('01-04-2021','dd-mm-yyyy'),2);

commit;

```

select *
from REZERVARE;

Script Output x Query Result x

SQL | All Rows Fetched: 14 in 0.001 seconds

	COD_REZERVARE	DATA_REZERVARE	NR_PERSOANE
1	700	02-APR-21	2
2	701	02-MAY-21	1
3	702	05-JAN-21	2
4	703	23-FEB-21	4
5	704	19-MAY-21	2
6	705	14-MAR-21	2
7	706	25-JAN-21	3
8	707	13-JAN-21	150
9	708	21-JAN-21	200
10	709	10-FEB-21	90
11	710	16-JAN-21	90
12	711	28-JAN-21	95
13	712	28-APR-21	5
14	713	01-APR-21	2

```

create sequence secventa_comanda_id
start with 500
increment by 1
maxvalue 9999
nocycle
nocache;

insert into COMANDA
values (secventa_comanda_id.nextval,'card',null,to_date('23-04-2021','dd-mm-yyyy'),101,15);

insert into COMANDA
values (secventa_comanda_id.nextval,'cash',null,to_date('10-04-2021','dd-mm-yyyy'),101,15);

```


insert into COMANDA

values (secventa_comanda_id.nextval,'card',null,to_date('13-05-2021','dd-mm-yyyy'),106,28);

insert into COMANDA

values (secventa_comanda_id.nextval,'card',null,to_date('09-01-2021','dd-mm-yyyy'),102,25);

insert into COMANDA

values (secventa_comanda_id.nextval,'cash',null,to_date('01-02-2021','dd-mm-yyyy'),103,32);

insert into COMANDA

values (secventa_comanda_id.nextval,'card',null,to_date('19-04-2021','dd-mm-yyyy'),105,30);

insert into COMANDA

values (secventa_comanda_id.nextval,'cash',null,to_date('03-04-2021','dd-mm-yyyy'),104,20);

commit;

COD_COMANDA	TIP_PLATA	VALOARE	DATA_COMANDA	COD_RESTAURANT	COD_ANGAJAT
1	500 card	(null)	23-APR-21	101	15
2	501 cash	(null)	10-APR-21	101	15
3	502 card	(null)	13-MAY-21	106	28
4	503 card	(null)	09-JAN-21	102	25
5	504 cash	(null)	01-FEB-21	103	32
6	505 card	(null)	19-APR-21	105	30
7	506 cash	(null)	03-APR-21	104	20

insert into CLIENT

values (40,'Rapeanu','Adrian','0722334455',16,500);

insert into CLIENT

values (41,'Dragomir','Elena','0722334467',17,503);

insert into CLIENT

values (42,'Frusinoiu','Andrei','0700334467',19,501);

insert into CLIENT

values (43,'Mares','Catalin','0722330101',18,505);

insert into CLIENT

values (44,'Petre','Mihaela','0722339898',null,null);

insert into CLIENT

values (45,'Sorescu','Ana','0766339898',null,null);

insert into CLIENT

values (46,'Soare','Mihaela','0755539898',null,null);

insert into CLIENT

```
values (47,'Ristoiu','Rares','0766777898',null,null);
```

```
insert into CLIENT
```

```
values (48,'Ciurea','Claudia','0722355558',null,null);
```

```
insert into CLIENT
```

```
values (49,'Vulpe','Alin','0765398078',null,null);
```

```
insert into CLIENT
```

```
values (50,'Negoita','Eduard','0765398009',null,null);
```

```
insert into CLIENT
```

```
values (51,'Neagu','David','0765398008',null,null);
```

```
commit;
```

select *
from client;

Script Output x Query Result x

SQL | All Rows Fetched: 12 in 0.002 seconds

	COD_CLIENT	NUME_CLIENT	PRENUME_CLIENT	TELEFON	COD_ADRESA	COD_COMANDA
1	40	Rapeanu	Adrian	0722334455	16	500
2	41	Dragomir	Elena	0722334467	17	503
3	42	Frusinoiu	Andrei	0700334467	19	501
4	43	Mares	Catalin	0722330101	18	505
5	44	Petre	Mihaela	0722339898	(null)	(null)
6	45	Sorescu	Ana	0766339898	(null)	(null)
7	46	Soare	Mihaela	0755539898	(null)	(null)
8	47	Ristoiu	Rares	0766777898	(null)	(null)
9	48	Ciurea	Claudia	0722355558	(null)	(null)
10	49	Vulpe	Alin	07653980786	(null)	(null)
11	50	Negoita	Eduard	07653980090	(null)	(null)
12	51	Neagu	David	07653980091	(null)	(null)

```
insert into FACE
```

```
values (44,709,300,102);
```

```
insert into FACE
```

```
values (44,705,303,104);
```

```
insert into FACE
```

```
values (45,702,303,102);
```

```
insert into FACE
```

```
values (46,708,305,101);
```

```
insert into FACE
```

```
values (47,707,306,103);
```

```
insert into FACE
```

```
values (48,700,303,106);
```

```
insert into FACE
```

```
values (48,710,304,104);
```

```
insert into FACE
```

```
values (47,703,302,106);
```

```
insert into FACE
```

```
values (45,704,302,106);
```

```
insert into FACE
```

```
values (44,706,302,105);
```

```
insert into FACE
```

```
values (41,712,303,104);
```

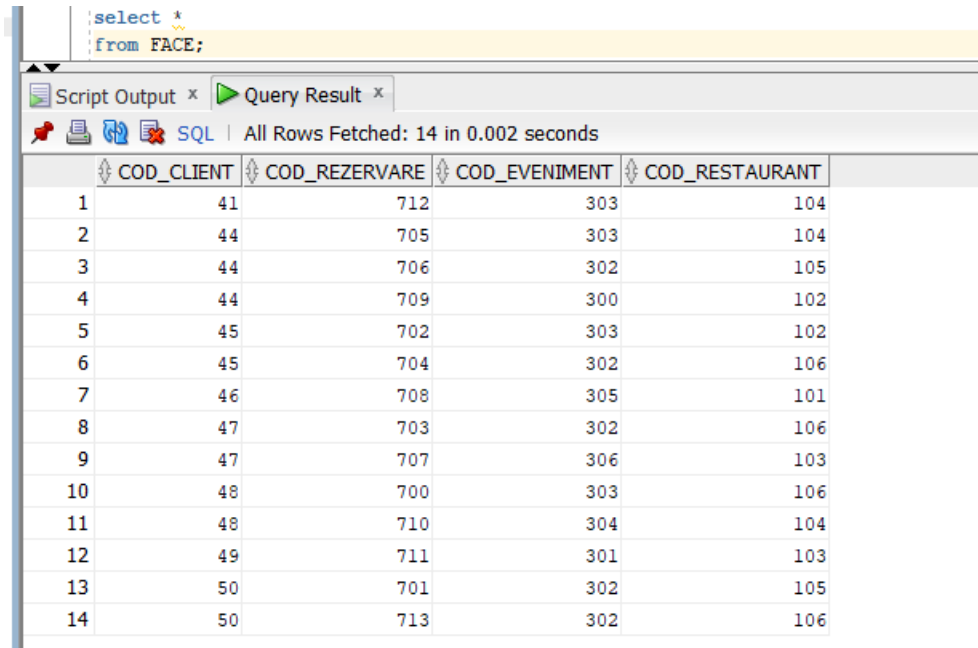
```
insert into FACE
```

```
values (50,713,302,106);
```

```
insert into FACE
values (49,711,301,103);
```

commit;

```
insert into FACE
values (50,701,302,105);
```



Script Output x Query Result x

SQL | All Rows Fetched: 14 in 0.002 seconds

	COD_CLIENT	COD_REZERVARE	COD_EVENTIMENT	COD_RESTAURANT
1	41	712	303	104
2	44	705	303	104
3	44	706	302	105
4	44	709	300	102
5	45	702	303	102
6	45	704	302	106
7	46	708	305	101
8	47	703	302	106
9	47	707	306	103
10	48	700	303	106
11	48	710	304	104
12	49	711	301	103
13	50	701	302	105
14	50	713	302	106

```
insert into ISTORIC_COMANDA
values (40,to_date('23-04-2021','dd-mm-yyyy'),10,500);
```

```
insert into ISTORIC_COMANDA
values (40,to_date('13-05-2021','dd-mm-yyyy'),9,502);
```

```
insert into ISTORIC_COMANDA
values (41,to_date('09-01-2021','dd-mm-yyyy'),10,503);
```

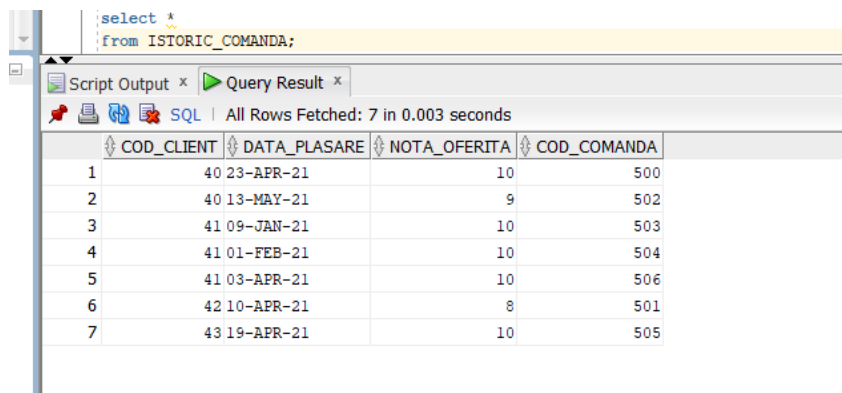
```
insert into ISTORIC_COMANDA
values (41,to_date('01-02-2021','dd-mm-yyyy'),10,504);
```

```
insert into ISTORIC_COMANDA
values (41,to_date('03-04-2021','dd-mm-yyyy'),10,506);
```

```
insert into ISTORIC_COMANDA
values (42,to_date('10-04-2021','dd-mm-yyyy'),8,501);
```

```
insert into ISTORIC_COMANDA
values (43,to_date('19-04-2021','dd-mm-yyyy'),10,505);
```

```
commit;
```



SQL | All Rows Fetched: 7 in 0.003 seconds

	COD_CLIENT	DATA_PLASARE	NOTA_OFERITA	COD_COMANDA
1	40	23-APR-21	10	500
2	40	13-MAY-21	9	502
3	41	09-JAN-21	10	503
4	41	01-FEB-21	10	504
5	41	03-APR-21	10	506
6	42	10-APR-21	8	501
7	43	19-APR-21	10	505

insert into PREPARAT
values (70,'ciorba de legume',10);

insert into PREPARAT
values (71,'ciorba de burta',15);

insert into PREPARAT
values (72,'pui la cuptor cu cartofi',23.5);

insert into PREPARAT
values (73,'paste cu ton',25);

insert into PREPARAT
values (74,'paste carbonara',26);

insert into PREPARAT
values (75,'pizza capriciosa',28);

insert into PREPARAT
values (76,'pizza casei',29);

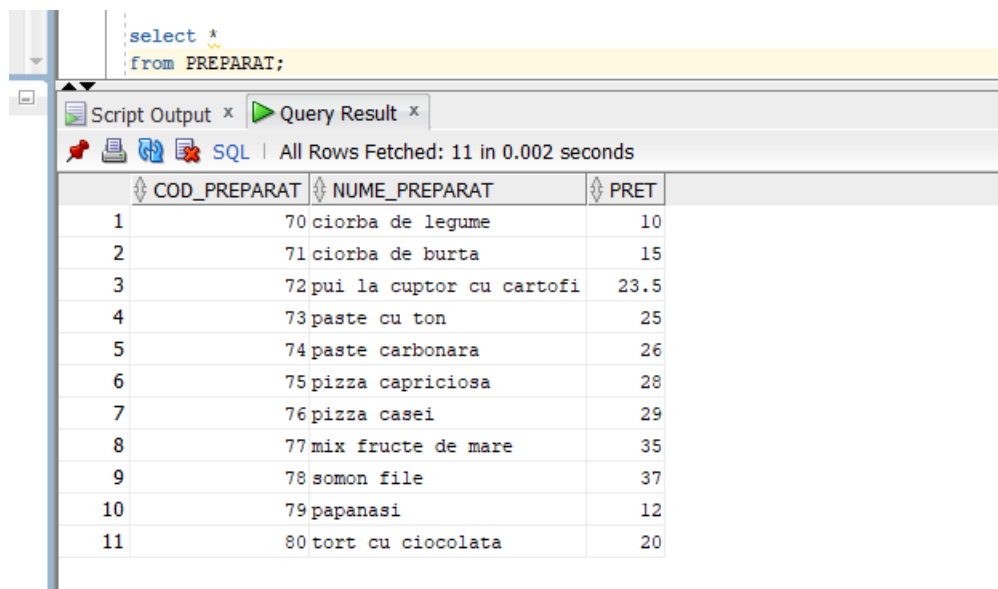
insert into PREPARAT
values (77,'mix fructe de mare',35);

insert into PREPARAT
values (78,'somon file',37);

insert into PREPARAT
values (79,'papanasi',12);

insert into PREPARAT
values (80,'tort cu ciocolata',20);

commit;



SQL | All Rows Fetched: 11 in 0.002 seconds

	COD_PREPARAT	NUME_PREPARAT	PRET
1	70	ciorba de legume	10
2	71	ciorba de burta	15
3	72	pui la cuptor cu cartofi	23.5
4	73	paste cu ton	25
5	74	paste carbonara	26
6	75	pizza capriciosa	28
7	76	pizza casei	29
8	77	mix fructe de mare	35
9	78	somon file	37
10	79	papanasi	12
11	80	tort cu ciocolata	20

insert into CUPRINDE
values (70,500,1);

insert into CUPRINDE
values (73,500,2);

insert into CUPRINDE
values (78,504,2);

insert into CUPRINDE
values (71,505,3);

```
insert into CUPRINDE
values (72,501,2);
```

```
insert into CUPRINDE
values (76,502,2);
```

```
insert into CUPRINDE
values (79,502,1);
```

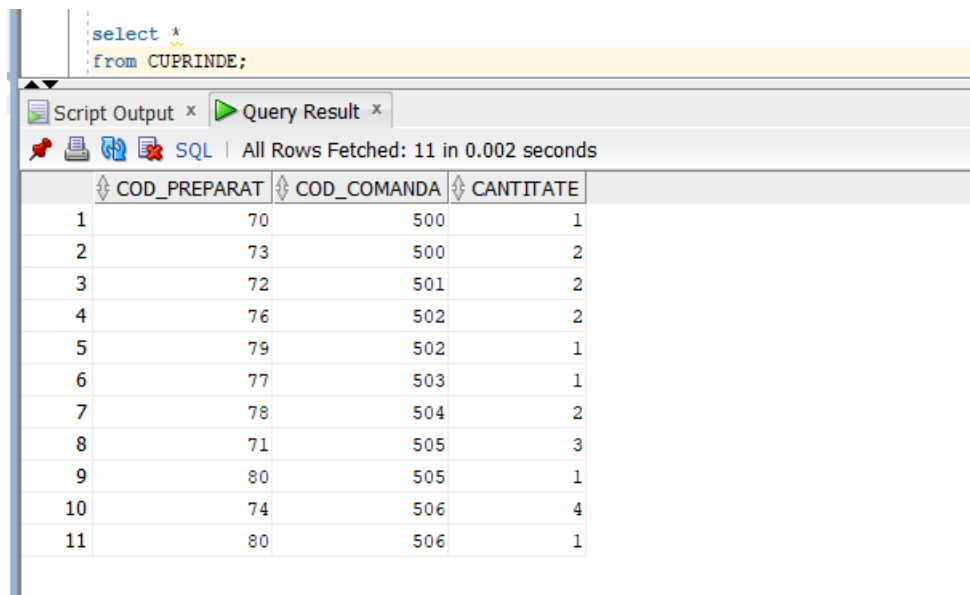
```
insert into CUPRINDE
values (77,503,1);
```

```
insert into CUPRINDE
values (80,505,1);
```

```
insert into CUPRINDE
values (74,506,4);
```

```
insert into CUPRINDE
values (80,506,1);
```

```
commit;
```



The screenshot shows a SQL query window with the command: `select * from CUPRINDE;`. Below the query, a 'Query Result' tab displays 11 rows of data. The columns are labeled COD_PREPARAT, COD_COMANDA, and CANTITATE. The data is as follows:

	COD_PREPARAT	COD_COMANDA	CANTITATE
1	70	500	1
2	73	500	2
3	72	501	2
4	76	502	2
5	79	502	1
6	77	503	1
7	78	504	2
8	71	505	3
9	80	505	1
10	74	506	4
11	80	506	1

6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze două tipuri de colecție studiate. Apelați subprogramul.

Cerință:

Definiți o procedură stocată care setează corespunzător valoarea fiecărei comenzi și afișează pe ecran codul comenzii, respectiv noua valoare setată. Se vor folosi colecții pentru a reține datele necesare rezolvării exercițiului. Apelați procedura.

```
CREATE OR REPLACE PROCEDURE ex6
IS
```

```
    TYPE tab_imb IS TABLE OF NUMBER;
```

```
    TYPE comanda_tab IS TABLE OF comanda.cod_comanda%TYPE INDEX BY
    BINARY_INTEGER;
```

```
    v_coduri_preparate tab_imb := tab_imb();
```

```
    tabel_comenzi comanda_tab;
```

```
    v_valoare comanda.valoare%TYPE;
```

```
    nr NUMBER;
```

```

    valoare_curenta comanda.valoare%TYPE;
BEGIN
    select cod_comanda
    bulk collect into tabel_comenzi
    from comanda;

    FOR i IN tabel_comenzi.FIRST..tabel_comenzi.LAST LOOP
        v_valoare := 0;

        select count(cod_preparat)
        into nr
        from cuprinde c
        where c.cod_comanda = tabel_comenzi(i);

        FOR j IN 1..nr LOOP
            v_coduri_preparate.EXTEND;
        END LOOP;

        select cod_preparat
        bulk collect into v_coduri_preparate
        from cuprinde c
        where c.cod_comanda= tabel_comenzi(i);

        FOR k IN v_coduri_preparate.FIRST..v_coduri_preparate.LAST LOOP
            select p.pret*c.cantitate
            into valoare_curenta
            from preparat p, cuprinde c
            where p.cod_preparat = v_coduri_preparate(k) and c.cod_preparat =
v_coduri_preparate(k) and c.cod_comanda = tabel_comenzi(i);

            v_valoare := v_valoare + valoare_curenta;
        END LOOP;

        update comanda
        set valoare = v_valoare
        where cod_comanda = tabel_comenzi(i);

        v_coduri_preparate.DELETE;
        dbms_output.put_line('Comanda ' || tabel_comenzi(i) || ' are valoarea de ' ||
v_valoare || ' lei.');
```

```

        dbms_output.put_line('Update realizat cu succes!');
        dbms_output.new_line();
    
```

END LOOP;
END ex6;

```
--6. Definiti o procedura stocata care seteaza corespunzator valoarea fiecărei comenzi si afiseaza pe ecran codul comenzii, respectiv noua valoare setata.
--Se vor folosi colectii pentru a retine datele necesare rezolvarii exercitiului. Apelati procedura.
CREATE OR REPLACE PROCEDURE ex6
IS
    TYPE tab_imb IS TABLE OF NUMBER;
    TYPE comanda_tab IS TABLE OF comanda.cod_comanda%TYPE INDEX BY BINARY_INTEGER;
    v_coduri_preparate tab_imb := tab_imb();
    tabel_comenzi comanda_tab;
    v_valoare comanda.valoare%type;
    nr NUMBER;
    valoare_curenta comanda.valoare%type;
BEGIN
    select cod_comanda
    bulk collect into tabel_comenzi
    from comanda;

    FOR i IN tabel_comenzi.FIRST..tabel_comenzi.LAST LOOP
        v_valoare := 0;

        select count(cod_preparat)
        into nr
        from cuprinde c
        where c.cod_comanda = tabel_comenzi(i);

        FOR j IN 1..nr LOOP
            v_coduri_preparate.EXTEND;
        END LOOP;

        select cod_preparat
        bulk collect into v_coduri_preparate
        from cuprinde c
        where c.cod_comanda = tabel_comenzi(i);

        v_valoare := v_valoare + v_coduri_preparate(nr);
    END LOOP;

    for i in tabel_comenzi.FIRST..tabel_comenzi.LAST LOOP
        update comanda set valoare = v_valoare where cod_comanda = tabel_comenzi(i);
    end loop;
END;
```

Script Output x
Task completed in 0.156 seconds

Procedure EX6 compiled

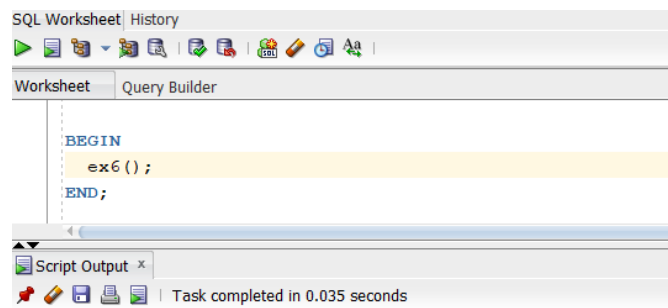
Inițial, toate comenzile au valoarea null deoarece așa sunt introduse în baza de date, urmând apoi să fie setate corespunzător în funcție de preparatele și cantitățile pe care le curpind.

```
select *
from comanda;
```

Script Output x Query Result x
SQL | All Rows Fetched: 7 in 0.003 seconds

	COD_COMANDA	TIP_PLATA	VALOARE	DATA_COMANDA	COD_RESTAURANT	COD_ANGAJAT
1	500	card	(null)	23-APR-21	101	15
2	501	cash	(null)	10-APR-21	101	15
3	502	card	(null)	13-MAI-21	106	28
4	503	card	(null)	09-IAN-21	102	25
5	504	cash	(null)	01-FEB-21	103	32
6	505	card	(null)	19-APR-21	105	30
7	506	cash	(null)	03-APR-21	104	20

BEGIN
ex6();
END;



Procedure EX6 compiled

Comanda 500 are valoarea de 60 lei.
Update realizat cu succes!

Comanda 501 are valoarea de 47 lei.
Update realizat cu succes!

Comanda 502 are valoarea de 70 lei.
Update realizat cu succes!

Comanda 503 are valoarea de 35 lei.
Update realizat cu succes!

Comanda 504 are valoarea de 74 lei.
Update realizat cu succes!

Comanda 505 are valoarea de 65 lei.
Update realizat cu succes!

Comanda 506 are valoarea de 124 lei.
Update realizat cu succes!

PL/SQL procedure successfully completed.

select *
from comanda;

Script Output x | Query Result x

SQL | All Rows Fetched: 7 in 0.101 seconds

	COD_COMANDA	TIP_PLATA	VALOARE	DATA_COMANDA	COD_RESTAURANT	COD_ANGAJAT
1	500	card	60	23-APR-21	101	15
2	501	cash	47	10-APR-21	101	15
3	502	card	70	13-MAI-21	106	28
4	503	card	35	09-IAN-21	102	25
5	504	cash	74	01-FEB-21	103	32
6	505	card	65	19-APR-21	105	30
7	506	cash	124	03-APR-21	104	20

Pentru rezolvarea problemei am folosit un tablou imbricat în care am reținut codurile preparatelor cuprinse într-o comanda și un tablou indexat în care am reținut toate codurile comenzilor. Parcurgând comenzile, am numărat câte preparate conține comanda respectivă pentru a apela extend pe tabloul imbricat ce

reține preparatele și am salvat lista acestor preparate. Parcurgând preparatele dintr-o comandă, am calculat valoarea curentă reprezentând produsul dintre prețul produsului respectiv și cantitatea sa, incrementând valoarea finală. La final am modificat valoarea comenzii și am șters tabloul imbricat cu preparatele curente.

7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.

Cerință:

Definiți o procedură stocată care pentru fiecare preparat afișează o listă cu detaliile comenzilor care cuprind preparatul respectiv. (codul comenzii, data la care a fost plasată comanda și tipul plății (cash/card)). Dacă preparatul nu este cuprins în nicio comandă, se va afișa un mesaj corespunzător. Apelați procedura.

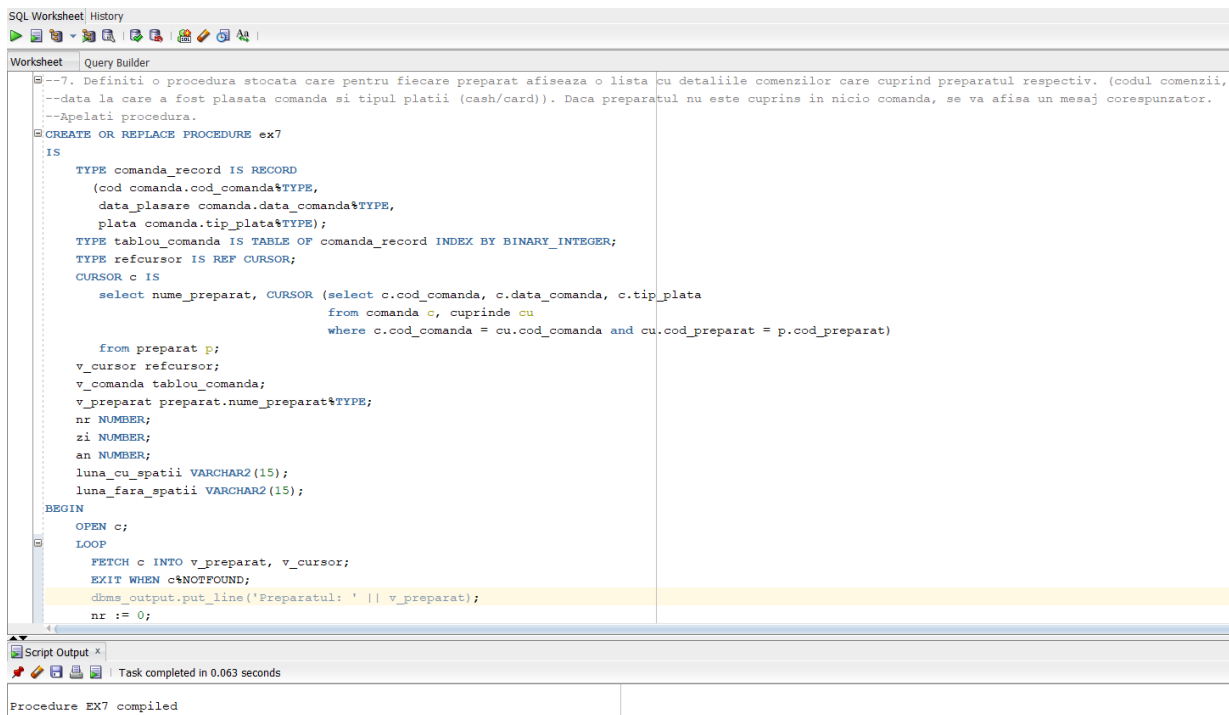
```
CREATE OR REPLACE PROCEDURE ex7
IS
  TYPE comanda_record IS RECORD
    (cod comanda.cod_comanda%TYPE,
     data_plasare comanda.data_comanda%TYPE,
     plata comanda.tip_plata%TYPE);
  TYPE tablou_comanda IS TABLE OF comanda_record INDEX BY
  BINARY_INTEGER;
  TYPE refcursor IS REF CURSOR;
  CURSOR c IS
    select nume_preparat, CURSOR (select c.cod_comanda, c.data_comanda, c.tip_plata
                                from comanda c, cuprinde cu
                                where c.cod_comanda = cu.cod_comanda and cu.cod_preparat =
p.cod_preparat)
    from preparat p;
  v_cursor refcursor;
  v_comanda tablou_comanda;
  v_preparat preparat.nume_preparat%TYPE;
  nr NUMBER;
  zi NUMBER;
  an NUMBER;
  luna_cu_spatii VARCHAR2(15);
  luna_fara_spatii VARCHAR2(15);
BEGIN
  OPEN c;
  LOOP
```

```

    FETCH c INTO v_preparat, v_cursor;
    EXIT WHEN c%NOTFOUND;
    dbms_output.put_line('Preparatul: ' || v_preparat);
    nr := 0;
    LOOP
        FETCH v_cursor BULK COLLECT INTO v_comanda;
        nr := v_comanda.COUNT;
        IF nr = 0 THEN
            dbms_output.put_line('Nu exista comenzi care sa cuprinda preparatul
respectiv!');
        ELSE
            dbms_output.put_line('Comenzile care cuprind preparatul ' || v_preparat || ':');
            FOR i IN v_comanda.FIRST..v_comanda.LAST LOOP
                zi := extract (day from v_comanda(i).data_plasare);

                select to_char(v_comanda(i).data_plasare, 'Month',
'NLS_DATE_LANGUAGE = Romanian')
                into luna_cu_spatii
                from dual;

                luna_fara_spatii := RTRIM(luna_cu_spatii, ' ');
                an := extract(year from v_comanda(i).data_plasare);
                dbms_output.put_line('Comanda cu codul ' || v_comanda(i).cod || ', plasata in
data de ' || zi || ' ' || luna_fara_spatii || ' ' || an
                || ' si care are tipul platii ' || v_comanda(i).plata || '.');
            END LOOP;
        END IF;
        EXIT WHEN v_cursor%NOTFOUND;
    END LOOP;
    dbms_output.new_line;
END LOOP;
CLOSE c;
END ex7;
```



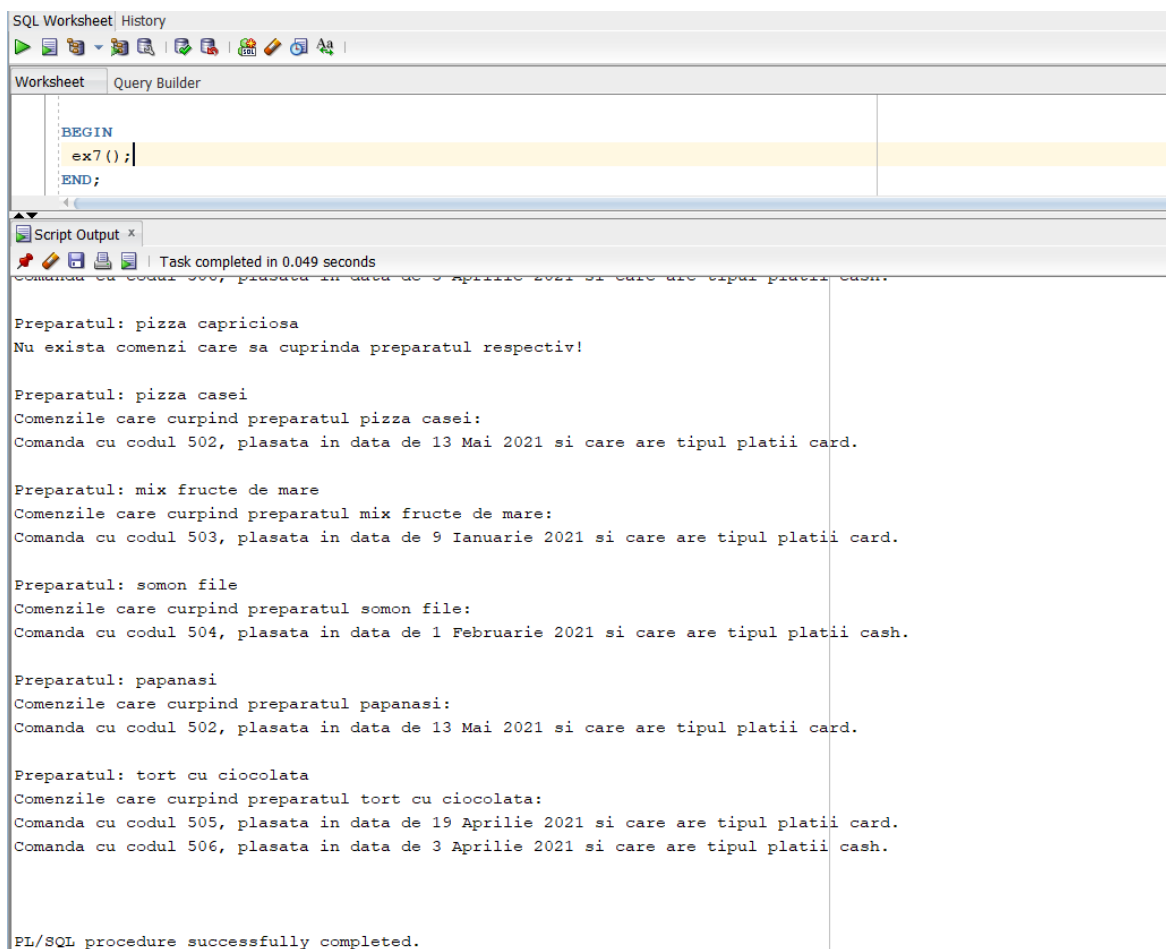
```

--7. Definiti o procedura stocata care pentru fiecare preparat afiseaza o lista cu detaliile comenzilor care cuprind preparatul respectiv. (codul comenzii,
--data la care a fost plasata comanda si tipul platii (cash/card)). Daca preparatul nu este cuprins in nicio comanda, se va afisa un mesaj corespunzator.
--Apelati procedura.
CREATE OR REPLACE PROCEDURE ex7
IS
    TYPE comanda_record IS RECORD
    (cod_comanda cod_comanda%TYPE,
    data_plasare comanda.data_comanda%TYPE,
    plata comanda.tip_plata%TYPE);
    TYPE tablou_comanda IS TABLE OF comanda_record INDEX BY BINARY_INTEGER;
    TYPE refcursor IS REF CURSOR;
    CURSOR c IS
        select nume_preparat, CURSOR (select c.cod_comanda, c.data_comanda, c.tip_plata
        from comanda c, cuprinde cu
        where c.cod_comanda = cu.cod_comanda and cu.cod_preparat = p.cod_preparat)
        from preparat p;
    v_cursor refcursor;
    v_comanda tablou_comanda;
    v_preparat preparat.nume_preparat%TYPE;
    nr NUMBER;
    zi NUMBER;
    an NUMBER;
    luna_cu_spatii VARCHAR2(15);
    luna_fara_spatii VARCHAR2(15);
BEGIN
    OPEN c;
    LOOP
        FETCH c INTO v_preparat, v_cursor;
        EXIT WHEN c%NOTFOUND;
        dbms_output.put_line('Preparatul: ' || v_preparat);
        nr := 0;
    
```

BEGIN

ex7();

END;



```

BEGIN
    ex7();
END;

```

Task completed in 0.049 seconds

Comanda cu codul 507, plasata in data de 9 Aprilie 2021 si care are tipul platii cash.

Preparatul: pizza capriciosa
Nu exista comenzi care sa cuprinda preparatul respectiv!

Preparatul: pizza casei
Comenzile care cuprind preparatul pizza casei:
Comanda cu codul 502, plasata in data de 13 Mai 2021 si care are tipul platii card.

Preparatul: mix fructe de mare
Comenzile care cuprind preparatul mix fructe de mare:
Comanda cu codul 503, plasata in data de 9 Ianuarie 2021 si care are tipul platii card.

Preparatul: somon file
Comenzile care cuprind preparatul somon file:
Comanda cu codul 504, plasata in data de 1 Februarie 2021 si care are tipul platii cash.

Preparatul: papanasi
Comenzile care cuprind preparatul papanasi:
Comanda cu codul 502, plasata in data de 13 Mai 2021 si care are tipul platii card.

Preparatul: tort cu ciocolata
Comenzile care cuprind preparatul tort cu ciocolata:
Comanda cu codul 505, plasata in data de 19 Aprilie 2021 si care are tipul platii card.
Comanda cu codul 506, plasata in data de 3 Aprilie 2021 si care are tipul platii cash.

PL/SQL procedure successfully completed.

Pentru rezolvarea cerinței am folosit expresii cursor (cursor imbricat/nested cursor). Acesta se declară, se deschide (open), se încarcă datele (fetch) și se închide (close). Pentru a putea extrage luna din data comenzii în limba română am folosit 'NLS_DATE_LANGUAGE = Romanian', iar pentru a șterge spațiile de la finalul denumirii lunii am folosit RTRIM.

8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un **subprogram stocat de tip funcție** care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Cerință:

Definiți o funcție stocată care să returneze un mesaj corespunzător cu un top 3. Funcția primește trei nume ale unor clienți, un nume de restaurant și o dată și face clasamentul clienților după numărul de comenzi plasate în restaurantul cu numele transmis, după data indicată. Tratați excepțiile care pot apărea. Apelați funcția astfel încât să evidențiați aceste excepții.

```
CREATE OR REPLACE FUNCTION ex8(
    nume1 client.nume_client%TYPE, nume2 client.nume_client%TYPE, nume3
client.nume_client%TYPE, nume_res restaurant.nume_restaurant%TYPE,
    data_com comanda.data_comanda%TYPE)
    RETURN VARCHAR2
IS
    v_cod_res NUMBER;
    v_cod_cl1 client.cod_client%TYPE;
    v_cod_cl2 client.cod_client%TYPE;
    v_cod_cl3 client.cod_client%TYPE;
    nr_cl1 NUMBER;
    nr_cl2 NUMBER;
    nr_cl3 NUMBER;
    aux NUMBER;
    aux_nume client.nume_client%TYPE;
    nume1_copie client.nume_client%TYPE;
    nume2_copie client.nume_client%TYPE;
    nume3_copie client.nume_client%TYPE;
    mesaj VARCHAR2(200);
BEGIN
    select cod_client
    into v_cod_cl1
    from client
```

```
where upper(ume_client) = upper(ume1);
```

```
select cod_client  
into v_cod_cl2  
from client  
where upper(ume_client) = upper(ume2);
```

```
select cod_client  
into v_cod_cl3  
from client  
where upper(ume_client) = upper(ume3);
```

```
select cod_restaurant  
into v_cod_res  
from restaurant  
where upper(ume_restaurant) = upper(ume_res);
```

```
select count(*)  
into nr_cl1  
from istoric_comanda i, client c, comanda co, restaurant r  
where upper(c.ume_client) = upper(ume1) and c.cod_client = i.cod_client and  
i.cod_comanda = co.cod_comanda and co.cod_restaurant = r.cod_restaurant  
and upper(r.ume_restaurant) = upper(ume_res) and co.data_comanda > data_com;
```

```
select count(*)  
into nr_cl2  
from istoric_comanda i, client c, comanda co, restaurant r  
where upper(c.ume_client) = upper(ume2) and c.cod_client = i.cod_client and  
i.cod_comanda = co.cod_comanda and co.cod_restaurant = r.cod_restaurant  
and upper(r.ume_restaurant) = upper(ume_res) and co.data_comanda > data_com;
```

```
select count(*)  
into nr_cl3  
from istoric_comanda i, client c, comanda co, restaurant r  
where upper(c.ume_client) = upper(ume3) and c.cod_client = i.cod_client and  
i.cod_comanda = co.cod_comanda and co.cod_restaurant = r.cod_restaurant  
and upper(r.ume_restaurant) = upper(ume_res) and co.data_comanda > data_com;
```

```
ume1_copie := ume1;  
ume2_copie := ume2;  
ume3_copie := ume3;
```

```

IF (nr_cl1 < nr_cl2) THEN
    aux := nr_cl1;
    nr_cl1 := nr_cl2;
    nr_cl2 := aux;
    aux_nume := nume1_copie;
    nume1_copie := nume2_copie;
    nume2_copie := aux_nume;
END IF;

```

```

IF (nr_cl1 < nr_cl3) THEN
    aux := nr_cl1;
    nr_cl1 := nr_cl3;
    nr_cl3 := aux;
    aux_nume := nume1_copie;
    nume1_copie := nume3_copie;
    nume3_copie := aux_nume;
END IF;

```

```

IF (nr_cl2 < nr_cl3) THEN
    aux := nr_cl2;
    nr_cl2 := nr_cl3;
    nr_cl3 := aux;
    aux_nume := nume2_copie;
    nume2_copie := nume3_copie;
    nume3_copie := aux_nume;
END IF;

```

```

mesaj := 'Top 3 dupa numarul de comenzi plasate in restaurantul ' || initcap(ume_res)
|| ' dupa data de ' || data_com || ': ' || nr_cl1
|| ' - ' || initcap(ume1_copie) || '; ' || nr_cl2 || ' - ' || initcap(ume2_copie) || '; ' || nr_cl3 || '
- ' || initcap(ume3_copie);
RETURN mesaj;

```

EXCEPTION

WHEN NO_DATA_FOUND THEN

IF v_cod_cl1 IS NULL THEN

mesaj := 'Nu exista client cu numele ' || initcap(ume1) || '!';

ELSIF v_cod_cl2 IS NULL THEN

mesaj := 'Nu exista client cu numele ' || initcap(ume2) || '!';

ELSIF v_cod_cl3 IS NULL THEN

mesaj := 'Nu exista client cu numele ' || initcap(ume3) || '!';

ELSE

```

        mesaj := 'Nu exista restaurant cu numele ' || initcap(ume_res) || '!';
    END IF;
    RETURN mesaj;
WHEN TOO_MANY_ROWS THEN
    IF v_cod_cl1 IS NULL THEN
        mesaj := 'Exista mai multi clienti cu numele ' || initcap(ume1) || '!';
    ELSIF v_cod_cl2 IS NULL THEN
        mesaj := 'Exista mai multi clienti cu numele ' || initcap(ume2) || '!';
    ELSIF v_cod_cl3 IS NULL THEN
        mesaj := 'Exista mai multi clienti cu numele ' || initcap(ume3) || '!';
    ELSE
        mesaj := 'Exista mai multe restaurante cu numele ' || initcap(ume_res) || '!';
    END IF;
    RETURN mesaj;
WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END ex8;

```

SQL Worksheet: History

0.047 seconds

Worksheet Query Builder

```

--8. Definiti o functie stocata care sa returneze un mesaj corespunzator cu un top 3. Functia primeste trei nume ale unor clienti, un nume de restaurant si o data
--si face clasamentul clientilor dupa numarul de comenzi plasate in restaurantul cu numele transmis, dupa data indicata.
--Tratati exceptiile care pot aparea. Apelati functia astfel incat sa evidentiati aceste exceptii.

CREATE OR REPLACE FUNCTION ex8(
    ume1 client.ume_client%TYPE, ume2 client.ume_client%TYPE, ume3 client.ume_client%TYPE, ume_res restaurant.ume_restaurant%TYPE,
    data_com comanda.data_comanda%TYPE)
    RETURN VARCHAR2
IS
    v_cod_res NUMBER;
    v_cod_cl1 client.cod_client%TYPE;
    v_cod_cl2 client.cod_client%TYPE;
    v_cod_cl3 client.cod_client%TYPE;
    nr_cl1 NUMBER;
    nr_cl2 NUMBER;
    nr_cl3 NUMBER;
    aux NUMBER;
    aux_ume client.ume_client%TYPE;
    ume1_copie client.ume_client%TYPE;
    ume2_copie client.ume_client%TYPE;
    ume3_copie client.ume_client%TYPE;
    mesaj VARCHAR2(200);
BEGIN
    select cod_client
    into v_cod_cl1
    from client
    where upper(ume_client) = upper(ume1);

    select cod_client
    into v_cod_cl2
    from client
    where upper(ume_client) = upper(ume2);

```

Script Output

Task completed in 0.047 seconds

Function EX8 compiled

- Exemple apeluri care nu generează excepții:

```

BEGIN
dbms_output.put_line(ex8('rapeanu', 'dragomir', 'Frusinoiu', 'lazarini', to_date('02-02-
2020', 'dd-mm-yyyy')));
END;

```

```
BEGIN
dbms_output.put_line(ex8('rapeanu', 'dragomir', 'Frusinoiu', 'lazarini', to_date('02-02-2020', 'dd-mm-yyyy')));
END;
```

Script Output x
Task completed in 0.081 seconds

Top 3 dupa numarul de comenzi plasate in restaurantul Lazarini dupa data de 02-FEB-20: 1 - Dragomir; 0 - Rapeanu; 0 - Frusinoiu

PL/SQL procedure successfully completed.

```
BEGIN
dbms_output.put_line(ex8('rapeanu', 'dragomir', 'Frusinoiu', 'da vinci', to_date('02-02-2020', 'dd-mm-yyyy')));
END;
```

```
BEGIN
dbms_output.put_line(ex8('rapeanu', 'dragomir', 'Frusinoiu', 'da vinci', to_date('02-02-2020', 'dd-mm-yyyy')));
END;
```

Script Output x
Task completed in 0.054 seconds

Top 3 dupa numarul de comenzi plasate in restaurantul Da Vinci dupa data de 02-FEB-20: 1 - Rapeanu; 1 - Frusinoiu; 0 - Dragomir

PL/SQL procedure successfully completed.

- Exemplu apel care generează excepția NO_DATA_FOUND și afișează mesajul corespunzător:

```
BEGIN
dbms_output.put_line(ex8('negoit', 'dragomir', 'Frusinoiu', 'da vinci', to_date('02-02-2020', 'dd-mm-yyyy')));
END;
```

```
BEGIN
dbms_output.put_line(ex8('negoit', 'dragomir', 'Frusinoiu', 'da vinci', to_date('02-02-2020', 'dd-mm-yyyy')));
END;
```

Script Output x
Task completed in 0.047 seconds

Nu exista client cu numele Negoit!

PL/SQL procedure successfully completed.

```
BEGIN
dbms_output.put_line(ex8('negoita', 'dragomir', 'Frusinoiu', 'da vinci', to_date('02-02-2020', 'dd-mm-yyyy')));
END;
```



```

BEGIN
dbms_output.put_line(ex8('negoita', 'dragomir', 'Frusinoiu', 'da vincii', to_date('02-02-2020', 'dd-mm-yyyy')));
END;

```

Script Output x
Task completed in 0.042 seconds

Nu exista restaurant cu numele Da Vincii!

PL/SQL procedure successfully completed.

- Exemplu apel care generează excepția TOO_MANY_ROWS și afișează mesajul corespunzător:

```

BEGIN
dbms_output.put_line(ex8('negoita', 'neagu', 'Frusinoiu', 'da vinci', to_date('02-02-2020', 'dd-mm-yyyy')));
END;

```

```

BEGIN
dbms_output.put_line(ex8('negoita', 'neagu', 'Frusinoiu', 'da vinci', to_date('02-02-2020', 'dd-mm-yyyy')));
END;

```

Script Output x
Task completed in 0.036 seconds

Exista mai multi clienti cu numele Neagu!

PL/SQL procedure successfully completed.

Pentru rezolvarea problemei, inițial am salvat codurile corespunzătoare parametrilor, iar apoi am numărat comenzile care corespund cerinței. Am făcut acest lucru deoarece, dacă nu aș fi salvat mai întâi codul, excepția NO_DATA_FOUND nu s-ar fi generat niciodată, returnându-se doar valoarea 0. Pentru a face corect clasamentul, am comparat cele 3 valori interschimbându-le dacă era cazul (interschimbând de asemenea și numele clienților pentru a putea afișa corect mesajul final). Am realizat câte o copie a numelor deoarece nu se poate modifica direct parametrul primit în funcție.

9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO_DATA_FOUND și TOO_MANY_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Cerință:

Definiți o procedură stocată care, pentru un nume al unui client transmis ca parametru și o opțiune, realizează următoarele:

- Opțiunea 1: Afișează o listă cu detaliile rezervărilor făcute de clientul respectiv (număr de persoane, data rezervării, data evenimentului, restaurantul în care are loc evenimentul și tipul evenimentului);
- Opțiunea 2: Afișează o listă cu detaliile comenzilor făcute de clientul respectiv (cod comandă, valoarea comenzii și tipul plății);
- Opțiunea 3: Răspunde la întrebarea: “Există comenzi care au valoarea un număr divizibil cu 10 pe care clientul transmis ca parametru să nu le fi făcut?”.

Se vor trata excepțiile care pot apărea. Apelați procedura astfel încât să se evidențieze excepțiile tratate.

```
CREATE OR REPLACE PROCEDURE ex9
```

```
(nume client.nume_client%TYPE, optiune NUMBER)
```

```
IS
```

```
TYPE informatii IS RECORD
```

```
(nr_persoane rezervare.nr_persoane%TYPE,  
data_r rezervare.data_rezervare%TYPE,  
data_e eveniment.data_eveniment%TYPE,  
nume_res restaurant.nume_restaurant%TYPE,  
tip_e tip_eveniment.descriere%TYPE);  
v_inf informatii;
```

```
CURSOR c(parametru client.nume_client%type) IS
```

```
select r.nr_persoane, r.data_rezervare , e.data_eveniment, res.nume_restaurant,  
t.descriere
```

```
from client c, REZERVARE r, EVENIMENT e, RESTAURANT res, FACE f,  
tip_eveniment t
```

```
where f.cod_rezervare=r.cod_rezervare and f.cod_restaurant=res.cod_restaurant  
and f.cod_eveniment=e.cod_eveniment and e.cod_tip = t.cod_tip and f.cod_client =  
c.cod_client and upper(c.nume_client) = upper(parametru);
```

```
TYPE comanda_tip IS REF CURSOR RETURN comanda%ROWTYPE;
```

```
v_comanda comanda_tip;
```

```
v_com comanda%ROWTYPE;
```

```
contor NUMBER :=1;
```

```
ok NUMBER := 0;
```

```
TYPE clienti_tab IS TABLE OF istoric_comanda.cod_client%TYPE INDEX BY  
BINARY_INTEGER;
```

```
coduri_clienti clienti_tab;
```

```

v_cod_client client.cod_client%TYPE;
BEGIN
  select cod_client
  into v_cod_client
  from client
  where upper(ume_client) = upper(ume);

  IF optiune = 1 THEN
    dbms_output.put_line('Rezervari pentru clientul ' || initcap(ume) || ':');
    OPEN c(ume);
    LOOP
      FETCH c INTO v_inf;
      EXIT WHEN c%NOTFOUND;
      dbms_output.put_line(contor || '. Numar persoane: ' || v_inf.nr_persoane || ', Data
rezervare: ' || v_inf.data_r || ', Data eveniment: ' || v_inf.data_e || ', Nume restaurant: ' ||
v_inf.ume_res || ', Tip eveniment: ' || v_inf.tip_e);
      contor := contor + 1;
    END LOOP;
    CLOSE c;
  ELSIF optiune = 2 THEN
    dbms_output.put_line('Comenzi pentru clientul ' || initcap(ume) || ':');
    OPEN v_comanda FOR select co.*
                        from comanda co, client c, istoric_comanda i
                        where upper(c.ume_client) = upper(ume) and
c.cod_client = i.cod_client and i.cod_comanda = co.cod_comanda;
    LOOP
      FETCH v_comanda INTO v_com;
      EXIT WHEN v_comanda%NOTFOUND;
      dbms_output.put_line(contor || '. Comanda ' || v_com.cod_comanda || ' cu valoarea
de ' || v_com.valoare || ' lei, cu tipul platii ' || v_com.tip_plata);
      contor := contor + 1;
    END LOOP;
    CLOSE v_comanda;
  ELSIF optiune = 3 THEN
    select distinct(a.cod_client)
    bulk collect into coduri_clienti
    from istoric_comanda a
    where not exists (select 1
                     from COMANDA c
                     where mod(c.valoare, 10) = 0 and not exists (select 1
                     from istoric_comanda b

```

```

                                where c.cod_comanda=b.cod_comanda
and b.cod_client=a.cod_client));
    FOR i IN coduri_clienti.FIRST..coduri_clienti.LAST LOOP
        IF coduri_clienti(i) = v_cod_client THEN
            ok := 1;
        END IF;
    END LOOP;
    IF ok = 1 THEN
        dbms_output.put_line('Pentru ' || initcap(ume) || ' nu exista comenzi cu valoarea
un numar divizibil cu 10 pe care sa nu le fi facut.');
```

ELSE

```

        dbms_output.put_line('Pentru ' || initcap(ume) || ' exista comenzi cu valoarea un
numar divizibil cu 10 pe care sa nu le fi facut.');
```

END IF;

ELSE

```

        dbms_output.put_line('Optiune invalida!');
```

END IF;

EXCEPTION

WHEN NO_DATA_FOUND THEN

```

    RAISE_APPLICATION_ERROR(-20000,'Nu exista clientul cu numele transmis ca
parametru!');
```

WHEN TOO_MANY_ROWS THEN

```

    RAISE_APPLICATION_ERROR(-20001,'Exista mai multi clienti cu numele
transmis ca parametru!');
```

WHEN OTHERS THEN

```

    RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
```

END ex9;

```

--9. Definiti o procedura stocata care, pentru un nume al unui client transmis ca parametru si o optiune, realizeaza urmatoarele:
-- Optiunea 1: Afiseaza o lista cu detaliile rezervarilor facute de clientul respectiv (numar de persoane, data rezervarii, data evenimentului, restaurantul
-- in care are loc evenimentul si tipul evenimentului);
-- Optiunea 2: Afiseaza o lista cu detaliile comenzilor facute de clientul respectiv (cod comanda, valoarea comenzii si tipul platii);
-- Optiunea 3: Raspunde la intrebarea: 'Exista comenzi care au valoarea un numar divizibil cu 10 pe care clientul transmis ca parametru sa nu le fi facut?'.
-- Se vor trata exceptiile care pot aparea.
-- Apelati procedura astfel incat sa se evedentieze exceptiile tratate.
CREATE OR REPLACE PROCEDURE ex9
(nume_client.nume_client%TYPE, optiune NUMBER)
IS
    TYPE informatii IS RECORD
    (nr_persoane rezervare.nr_persoane%TYPE,
    data_r rezervare.data_rezervare%TYPE,
    data_e eveniment.data_eveniment%TYPE,
    nume_res restaurant.nume_restaurant%TYPE,
    tip_e tip_eveniment.descriere%TYPE);
    v_inf informatii;

    CURSOR c(parametru client.nume_client%type) IS
    select r.nr_persoane, r.data_rezervare, e.data_eveniment, res.nume_restaurant, t.descriere
    from client c, REZERVARE r, EVENIMENT e, RESTAURANT res, FACE f, tip_eveniment t
    where f.cod_rezervare=r.cod_rezervare and f.cod_restaurant=res.cod_restaurant
    and f.cod_eveniment=e.cod_eveniment and e.cod_tip = t.cod_tip and f.cod_client = c.cod_client and upper(c.nume_client) = upper(parametru);

    TYPE comanda_tip IS REF CURSOR RETURN comanda%ROWTYPE;
    v_comanda comanda_tip;
    v_cod_comanda%TYPE;

```

Script Output x

Task completed in 0.083 seconds

Procedure EX9 compiled

- Exemple apeluri care nu generează excepții:

execute ex9('negoita', 1);

```

execute ex9('negoita', 1);

```

Script Output x

Task completed in 0.034 seconds

Procedure EX9 compiled

Rezervari pentru clientul Negoita:

1. Numar persoane: 1, Data rezervare: 02-MAY-21, Data eveniment: 17-JUL-21, Nume restaurant: Trattoria, Tip eveniment: Petreceri Weekend
2. Numar persoane: 2, Data rezervare: 01-APR-21, Data eveniment: 17-JUL-21, Nume restaurant: Toscana, Tip eveniment: Petreceri Weekend

PL/SQL procedure successfully completed.

execute ex9('dragomir', 2);

```

execute ex9('dragomir', 2);

```

Script Output x

Task completed in 0.04 seconds

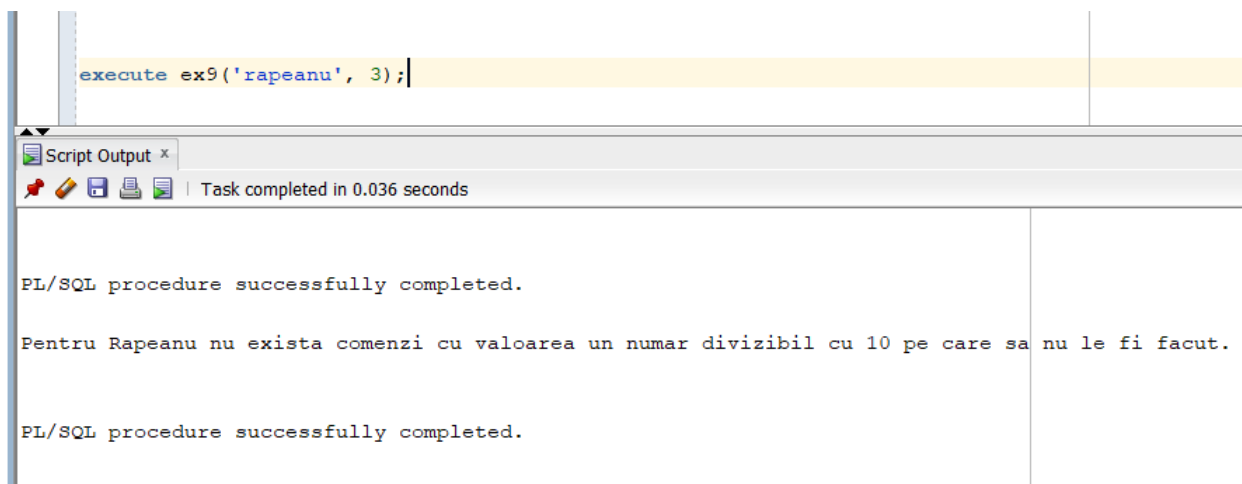
PL/SQL procedure successfully completed.

Comenzi pentru clientul Dragomir:

1. Comanda 503 cu valoarea de 35 lei, cu tipul platii card
2. Comanda 504 cu valoarea de 74 lei, cu tipul platii cash
3. Comanda 506 cu valoarea de 124 lei, cu tipul platii cash

PL/SQL procedure successfully completed.

execute ex9('rapeanu', 3);



The screenshot shows a SQL Developer window with a script titled 'execute ex9('rapeanu', 3);'. Below the script, the 'Script Output' pane shows the execution results. The output consists of three lines: 'PL/SQL procedure successfully completed.', 'Pentru Rapeanu nu exista comenzi cu valoarea un numar divizibil cu 10 pe care sa nu le fi facut.', and 'PL/SQL procedure successfully completed.'. The task completed in 0.036 seconds.

```
execute ex9('rapeanu', 3);
```

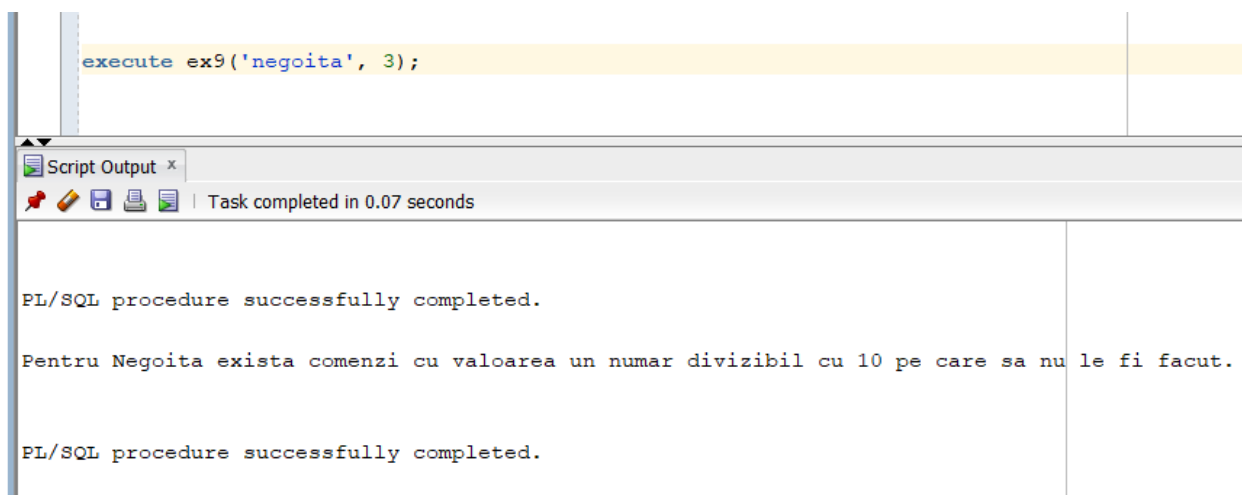
Script Output x
Task completed in 0.036 seconds

PL/SQL procedure successfully completed.

Pentru Rapeanu nu exista comenzi cu valoarea un numar divizibil cu 10 pe care sa nu le fi facut.

PL/SQL procedure successfully completed.

execute ex9('negoita', 3);



The screenshot shows a SQL Developer window with a script titled 'execute ex9('negoita', 3);'. Below the script, the 'Script Output' pane shows the execution results. The output consists of three lines: 'PL/SQL procedure successfully completed.', 'Pentru Negoita exista comenzi cu valoarea un numar divizibil cu 10 pe care sa nu le fi facut.', and 'PL/SQL procedure successfully completed.'. The task completed in 0.07 seconds.

```
execute ex9('negoita', 3);
```

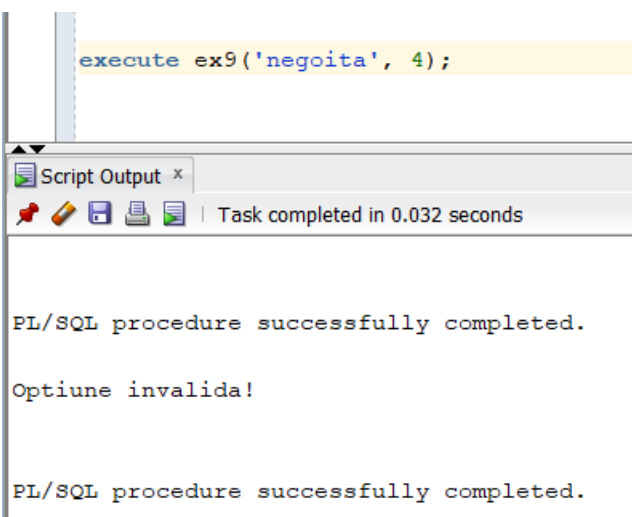
Script Output x
Task completed in 0.07 seconds

PL/SQL procedure successfully completed.

Pentru Negoita exista comenzi cu valoarea un numar divizibil cu 10 pe care sa nu le fi facut.

PL/SQL procedure successfully completed.

execute ex9('negoita', 4);



The screenshot shows a SQL Developer window with a script titled 'execute ex9('negoita', 4);'. Below the script, the 'Script Output' pane shows the execution results. The output consists of three lines: 'PL/SQL procedure successfully completed.', 'Optiune invalida!', and 'PL/SQL procedure successfully completed.'. The task completed in 0.032 seconds.

```
execute ex9('negoita', 4);
```

Script Output x
Task completed in 0.032 seconds

PL/SQL procedure successfully completed.

Optiune invalida!

PL/SQL procedure successfully completed.

- Exemple apeluri care generează excepții:

execute ex9('nume', 1);

```

execute ex9('nume', 1);

```

Script Output x
Task completed in 0.065 seconds

Error starting at line : 571 in command -
BEGIN ex9('nume', 1); END;
Error report -
ORA-20000: Nu exista clientul cu numele transmis ca parametru!
ORA-06512: at "ANDREEA_SORA1.EX9", line 77
ORA-06512: at line 1
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.

Pentru excepția TOO_MANY_ROWS am inserat încă un client în baza de date cu același nume cu un client deja existent:

insert into client

values (53, 'Neagu', 'Marian', '0766777890', null, null);

execute ex9('neagu', 1);

```

execute ex9('neagu', 1);

```

Script Output x
Task completed in 0.057 seconds

Error starting at line : 571 in command -
BEGIN ex9('neagu', 1); END;
Error report -
ORA-20001: Exista mai multi clienti cu numele transmis ca parametru!
ORA-06512: at "ANDREEA_SORA1.EX9", line 79
ORA-06512: at line 1

Pentru rezolvare, am folosit un record pentru a înregistra informațiile necesare unei rezervări, un cursor parametrizat care va obține lista rezervărilor pentru un client dat, un cursor dinamic pentru a reține detaliile comenzilor unui client dat și un tablou indexat în care am reținut codurile clienților care corespund opțiunii 3. Opțiunea 3 folosește operatorul DIVISION utilizând de două ori NOT EXISTS. Pentru început, rețin codurile clienților pentru care nu există o comandă cu valoarea un număr divizibil cu 10, pe care clientul respectiv să nu o fi făcut-o. Am folosit *select constantă* deoarece nu este necesar ca instrucțiunea să returneze o

anumită valoare și este mai eficient din punct de vedere al performanței.
Parcurgând acest tablou, dacă clientul căutat se află în tablou, rețin acest lucru și afișez un mesaj corespunzător.

10. Definiți un *trigger* de tip LMD la nivel de comandă. Declanșați *trigger*-ul.

Cerință:

Definiți un *trigger* LMD la nivel de comandă care nu permite modificarea tabelului preparat într-o zi de duminică.

De asemenea, se va respecta:

- a) Să nu se insereze mai mult de 100 de preparate deoarece în acest fel s-ar îngreuna pregătirea acestora, dacă meniul ar fi atât de variat. În plus, ar necesita prea multă materie primă.
- b) Să nu se șteargă preparate astfel încât să rămână mai puțin de 5.

Declanșați *trigger*-ul.

Pentru început, am modificat constrângerea cheii externe din tabelul CUPRINDE. Inițial nu s-a precizat ON DELETE CASCADE, iar acum, dacă dorim să testăm *trigger*-ul conform cerinței de mai sus, nu vom putea șterge preparate care apar în tabelul asociativ CUPRINDE. Vom face modificările necesare:

```
ALTER TABLE cuprinde  
DROP CONSTRAINT FK_CUP_PREP;
```

```
ALTER TABLE cuprinde  
ADD CONSTRAINT FK_CUP_PREP  
FOREIGN KEY (cod_preparat)  
REFERENCES PREPARAT(cod_preparat)  
ON DELETE CASCADE;
```

```
select constraint_name, search_condition, r_constraint_name, delete_rule  
from user_constraints  
where TABLE_NAME = 'CUPRINDE';
```


SQL Worksheet: History

Worksheet Query Builder

```
--10. Definiti un trigger IMD la nivel de comanda care nu permite modificarea tabelului preparat intr-o zi de duminica.
--De asemenea, se va respecta:
--a) Sa nu se insereze mai mult de 100 de preparate deoarece in acest fel s-ar ingreuna pregatirea acestora, daca meniul ar fi atat de variat. In plus, ar necesita
--prea multa materie prima.
--b) Sa nu se stearga preparate astfel incat sa ramana mai putin de 5.
--Declansati trigger-ul.
ALTER TABLE cuprinde
DROP CONSTRAINT FK_CUP_PREP;

ALTER TABLE cuprinde
ADD CONSTRAINT FK_CUP_PREP
FOREIGN KEY (cod_preparat)
REFERENCES PREPARAT(cod_preparat)
ON DELETE CASCADE;

select constraint_name, search_condition, r_constraint_name, delete_rule
from user_constraints
where TABLE_NAME = 'CUPRINDE';
```

Query Result

All Rows Fetched: 4 in 0.002 seconds

CONSTRAINT_NAME	SEARCH_CONDITION	R_CONSTRAINT_NAME	DELETE_RULE
1 FK_CUP_PREP	(null)	FK_PREPARAT	CASCADE
2 NULL_CANTITATE	"CANTITATE" IS NOT NULL	(null)	(null)
3 FK_CUPRINDE	(null)	(null)	(null)
4 FK_CUP_COM	(null)	FK_COMANDA	NO ACTION

CREATE OR REPLACE TRIGGER ex10

BEFORE UPDATE OR INSERT or DELETE on preparat

DECLARE

v_nr_preparate NUMBER;

v_zi VARCHAR2(20);

BEGIN

select to_char(sysdate, 'Day', 'NLS_DATE_LANGUAGE = Romanian')

into v_zi

from dual;

IF RTRIM(UPPER(v_zi)) = UPPER('duminica') THEN

RAISE_APPLICATION_ERROR(-20004, 'Nu se pot face modificari duminica!');

ELSE

select count(cod_preparat)

into v_nr_preparate

from preparat;

IF INSERTING AND v_nr_preparate >= 100 THEN

RAISE_APPLICATION_ERROR(-20001, 'Nu se pot insera mai mult de 100 de preparate!');

END IF;

IF DELETING AND v_nr_preparate <= 5 THEN

RAISE_APPLICATION_ERROR(-20002, 'Nu se pot sterge atatea preparate astfel incat sa ramanem cu mai putin de 5!');

END IF;

END IF;

END ex10;

```

CREATE OR REPLACE TRIGGER ex10
BEFORE UPDATE OR INSERT OR DELETE ON preparat
DECLARE
    v_nr_preparate NUMBER;
    v_zi VARCHAR2(20);
BEGIN
    select to_char(sysdate, 'Day', 'NLS_DATE_LANGUAGE = Romanian')
    into v_zi
    from dual;

    IF RTRIM(UPPER(v_zi)) = UPPER('duminica') THEN
        RAISE_APPLICATION_ERROR(-20004, 'Nu se pot face modificari duminica!');
    ELSE
        select count(cod_preparat)
        into v_nr_preparate
        from preparat;

        IF INSERTING AND v_nr_preparate >= 100 THEN
            RAISE_APPLICATION_ERROR(-20001, 'Nu se pot insera mai mult de 100 de preparate!');
        END IF;

        IF DELETING AND v_nr_preparate <= 5 THEN
            RAISE_APPLICATION_ERROR(-20002, 'Nu se pot sterge atatea preparate astfel incat sa ramanem cu mai putin de 5!');
        END IF;
    END IF;
END ex10;

```

Script Output x Query Result x

Task completed in 0.079 seconds

Trigger EX10 compiled

- Declanșare trigger pentru excepția în cazul ștergerii:

```

BEGIN
FOR i IN 73..80 LOOP
    delete from preparat
    where cod_preparat = i;
END LOOP;
END;

```

```

SQL Worksheet History
Worksheet Query Builder
RAISE_APPLICATION_ERROR(-20002, 'Nu se pot sterge mai mult de 100 de preparate. ');
END IF;

IF DELETING AND v_nr_preparate <= 5 THEN
    RAISE_APPLICATION_ERROR(-20002, 'Nu se pot sterge atatea preparate astfel incat sa ramanem cu mai putin de 5!');
END IF;
END IF;
END ex10;

BEGIN
FOR i IN 73..80 LOOP
    delete from preparat
    where cod_preparat = i;
END LOOP;
END;

Script Output x Query Result x
Task completed in 0.109 seconds

Trigger EX10 compiled

Error starting at line : 550 in command -
BEGIN
FOR i IN 73..80 LOOP
    delete from preparat
    where cod_preparat = i;
END LOOP;
END;
Error report -
ORA-20002: Nu se pot sterge atatea preparate astfel incat sa ramanem cu mai putin de 5!
ORA-06512: at "ANDREEA_SORA1.EX10", line 21
ORA-04088: error during execution of trigger 'ANDREEA_SORA1.EX10'
ORA-06512: at line 3

```

- Declanșare trigger pentru excepția în cazul inserării:

```

BEGIN
FOR i IN 500..600 LOOP
    insert into preparat
    values(i, 'nume', 89);
END LOOP;
END;

```

```

SQL Worksheet History
Worksheet Query Builder

BEGIN
  FOR i IN 73..80 LOOP
    delete from preparat
      where cod_preparat = i;
  END LOOP;
END;

BEGIN
  FOR i IN 500..600 LOOP
    insert into preparat
      values(i, 'nume', 89);
  END LOOP;
END;

```

Script Output x Query Result x

Task completed in 0.096 seconds

ORA-04088: error during execution of trigger 'ANDREEA_SORA1.EX10'

ORA-06512: at line 3

Error starting at line : 559 in command -

```

BEGIN
  FOR i IN 500..600 LOOP
    insert into preparat
      values(i, 'nume', 89);
  END LOOP;
END;

```

Error report -

ORA-20001: Nu se pot insera mai mult de 100 de preparate!

ORA-06512: at "ANDREEA_SORA1.EX10", line 17

ORA-04088: error during execution of trigger 'ANDREEA_SORA1.EX10'

ORA-06512: at line 3

- Declanșare trigger pentru excepția în cazul modificării tabelului într-o duminică:

```
update preparat
```

```
set pret = 70
```

```
where cod_preparat = 78;
```

```

update preparat
set pret = 70
where cod_preparat = 78;

```

Script Output x

Task completed in 0.045 seconds

Error starting at line : 567 in command -

```

update preparat
set pret = 70
where cod_preparat = 78

```

Error report -

ORA-20004: Nu se pot face modificari duminica!

ORA-06512: at "ANDREEA_SORA1.EX10", line 10

ORA-04088: error during execution of trigger 'ANDREEA_SORA1.EX10'

- Exemplu ștergere acceptată (ștergere automata și din tabelul asociativ):

```
BEGIN
FOR i IN 75..80 LOOP
    delete from preparat
    where cod_preparat = i;
END LOOP;
END;
```

The image shows two side-by-side screenshots of a SQL IDE. Both screenshots display the same SQL script: a loop deleting records from the 'preparat' table with IDs 75 to 80, followed by two select queries. The left screenshot shows the results of the first select query, which lists details for 5 food items. The right screenshot shows the results of the second select query, which lists reservation details for the same 5 items.

COD_PREPARAT	NUME_PREPARAT	PRET
1	70 ciorba de legume	10
2	71 ciorba de burta	15
3	72 pui la cuptor cu cartofi	23.5
4	73 paste cu ton	25
5	74 paste carbonara	26

COD_PREPARAT	COD_COMANDA	CANTITATE
1	70	500
2	73	500
3	72	501
4	71	505
5	74	506

11. Definiți un *trigger* de tip LMD la nivel de linie. Declanșați *trigger*-ul.

Cerință:

Definiți un *trigger* de tip LMD la nivel de linie care:

- Dacă este șters un client, va șterge din baza de date și istoricul său, cât și evidența rezervărilor la evenimente. Se va reține numărul de linii șterse.
- Dacă este schimbat codul unui client, se vor face modificările necesare în istoric cât și în evidența rezervărilor. Se va reține numărul de linii modificate.
- Dacă se inserează un nou client, se vor face următoarele verificări asupra numărului de telefon:

- numărul introdus să aibă 10 cifre;
- numărul introdus să înceapă cu '07' deoarece în baza de date vom dori doar numere de mobil, specifice țării noastre.

Declanșați *trigger*-ul.

```
CREATE PACKAGE pachet_contor IS
    v_nr_linii_update_face NUMBER;
```

```
v_nr_linii_update_istoric NUMBER;
v_nr_linii_delete_face NUMBER;
v_nr_linii_delete_istoric NUMBER;
END;

CREATE OR REPLACE TRIGGER ex11
BEFORE DELETE OR INSERT OR UPDATE on client
FOR EACH ROW
DECLARE
    exceptie1 EXCEPTION;
    exceptie2 EXCEPTION;
BEGIN
    IF DELETING THEN
        delete from face
        where cod_client = :OLD.cod_client;

        pachet_contor.v_nr_linii_delete_face := sql%rowcount;

        delete from istoric_comanda
        where cod_client = :OLD.cod_client;

        pachet_contor.v_nr_linii_delete_istoric := sql%rowcount;
    ELSIF UPDATING AND :OLD.cod_client != :NEW.cod_client THEN
        update face
        set cod_client = :NEW.cod_client
        where cod_client = :OLD.cod_client;

        pachet_contor.v_nr_linii_update_face := sql%rowcount;

        update istoric_comanda
        set cod_client = :NEW.cod_client
        where cod_client = :OLD.cod_client;

        pachet_contor.v_nr_linii_update_istoric := sql%rowcount;
    ELSIF INSERTING AND length(:NEW.telefon) != 10 THEN
        raise exceptie1;
    ELSIF INSERTING AND :NEW.telefon not like '07%' THEN
        raise exceptie2;
    END IF;
EXCEPTION
    WHEN exceptie1 THEN
```

```

RAISE_APPLICATION_ERROR (-20001, 'Numarul de telefon pe care doriti sa il
inserati nu este valid! Introduceti un numar de telefon mobil care are 10 cifre!');
WHEN exceptie2 THEN
    RAISE_APPLICATION_ERROR (-20002, 'Numarul de telefon pe care doriti sa il
inserati nu este valid! Introduceti un numar de telefon mobil care incepe cu 07!');
END ex11;

```

```

--11. Definiti un trigger de tip LMD la nivel de linie care:
--a) Daca este sters un client, va sterge din baza de date si istoricul sau, cat si evidenta rezervarilor la evenimente. Se va retine numarul de linii sterse.
--b) Daca este schimbat codul unui client, se vor face modificarile necesare in istoric cat si in evidenta rezervarilor. Se va retine numarul de linii modificate.
--c) Daca se insereaza un nou client, se vor face urmatoarele verificari asupra numarului de telefon:
--    - numarul introdus sa aiba 10 cifre;
--    - numarul introdus sa inceapa cu '07' deoarece in baza de date vom dori doar numere de mobil, specifice tarii noastre.
--Declansati trigger-ul.

CREATE PACKAGE pachet_contor IS
    v_nr_linii_update_face NUMBER;
    v_nr_linii_update_istoric NUMBER;
    v_nr_linii_delete_face NUMBER;
    v_nr_linii_delete_istoric NUMBER;
END;

CREATE OR REPLACE TRIGGER ex11
BEFORE DELETE OR INSERT OR UPDATE ON client
FOR EACH ROW
DECLARE
    exceptie1 EXCEPTION;
    exceptie2 EXCEPTION;
BEGIN
    IF DELETING THEN
        delete from face
        where cod_client = :OLD.cod_client;

        pachet_contor.v_nr_linii_delete_face := sql%rowcount;

        delete from istoric_comanda
        where cod_client = :OLD.cod_client;
    
```

Script Output x

Task completed in 0.1 seconds

Package PACHET_CONTOR compiled

Trigger EX11 compiled

- Declanșare trigger pentru cazul modificării în tabelul client:

```

DECLARE
    total NUMBER;
BEGIN
    update client
    set cod_client = 70
    where cod_client = 41;

    total := pachet_contor.v_nr_linii_update_face +
    pachet_contor.v_nr_linii_update_istoric;
    dbms_output.put_line('S-au modificat ' || total || ' linii.');
```

```
END;
```

```

DECLARE
    total NUMBER;
BEGIN
    update client
    set cod_client = 70
    where cod_client = 41;

    total := pachet_contor.v_nr_linii_update_face + pachet_contor.v_nr_linii_update_istoric;
    dbms_output.put_line('S-au modificat ' || total || ' linii.');
```

Script Output x | Task completed in 0.048 seconds

Trigger EX11 compiled

S-au modificat 4 linii.

PL/SQL procedure successfully completed.

Aceste 4 linii rezultă din: 3 linii modificate în istoric și o linie modificată în evidența rezervărilor.

```
select *
from istoric_comanda;
```

Script Output x | Query Result x | All Rows Fetched: 7 in 0.002 seconds

	COD_CLIENT	DATA_PLASARE	NOTA_OFERITA	COD_COMANDA
1	40	23-APR-21	10	500
2	40	13-MAY-21	9	502
3	70	09-JAN-21	10	503
4	70	01-FEB-21	10	504
5	70	03-APR-21	10	506
6	42	10-APR-21	8	501
7	43	19-APR-21	10	505

```
select *
from face;
```

Script Output x | Query Result x | All Rows Fetched: 14 in 0.002 seconds

	COD_CLIENT	COD_REZERVARE	COD_EVENTIMENT	COD_RESTAURANT
1	44	705	303	104
2	44	706	302	105
3	44	709	300	102
4	45	702	303	102
5	45	704	302	106
6	46	708	305	101
7	47	703	302	106
8	47	707	306	103
9	48	700	303	106
10	48	710	304	104
11	49	711	301	103
12	50	701	302	105
13	50	713	302	106
14	70	712	303	104

- Declanșare trigger pentru cazul ștergerii în tabelul client:

```
DECLARE
```

```
    total NUMBER;
```

```
BEGIN
```

```
    delete from client
```

```
    where cod_client = 70;
```

```
    total := pachet_contor.v_nr_linii_delete_face + pachet_contor.v_nr_linii_delete_istoric;
```

```
    dbms_output.put_line('S-au modificat ' || total || ' linii.');
```

```
END;
```



```

DECLARE
    total NUMBER;
BEGIN
    delete from client
    where cod_client = 70;

    total := pachet_contor.v_nr_linii_delete_face + pachet_contor.v_nr_linii_delete_istoric;
    dbms_output.put_line('S-au modificat ' || total || ' linii.');
```

Script Output x

Task completed in 0.052 seconds

S-au modificat 4 linii.

PL/SQL procedure successfully completed.

Aceste 4 linii rezultă din: 3 linii șterse din istoric și o linie ștearsă din evidența rezervărilor.

```
select *
from istoric_comanda;
```

Script Output x

Query Result x

SQL | All Rows Fetched: 4 in 0.002 seconds

COD_CLIENT	DATA_PLASARE	NOTA_OFERITA	COD_COMANDA
1	40 23-APR-21	10	500
2	40 13-MAY-21	9	502
3	42 10-APR-21	8	501
4	43 19-APR-21	10	505

```
select *
from face;
```

Script Output x

Query Result x

SQL | All Rows Fetched: 13 in 0.001 seconds

COD_CLIENT	COD_REZERVARE	COD_EVENTIMENT	COD_RESTAURANT	
1	44	705	303	104
2	44	706	302	105
3	44	709	300	102
4	45	702	303	102
5	45	704	302	106
6	46	708	305	101
7	47	703	302	106
8	47	707	306	103
9	48	700	303	106
10	48	710	304	104
11	49	711	301	103
12	50	701	302	105
13	50	713	302	106

- Declanșare trigger pentru cazul inserării unui client cu un număr de telefon care nu are 10 cifre:

```

insert into client
values (52, 'nume', 'prenume', '07983849469098', null, null);
```

```

insert into client
values (52, 'nume', 'prenume', '07983849469098', null, null);
```

Script Output x Query Result x

Task completed in 0.046 seconds

Error starting at line : 708 in command -

```

insert into client
values (52, 'nume', 'prenume', '07983849469098', null, null)
```

Error report -

ORA-20001: Numarul de telefon pe care doriti sa il inserati nu este valid! Introduceti un numar de telefon mobil care are 10 cifre!

ORA-06512: at "ANDREEA_SORA1.EX11", line 34

ORA-04088: error during execution of trigger 'ANDREEA_SORA1.EX11'

- Declanșare trigger pentru cazul inserării unui client cu un număr de telefon care nu începe cu '07':

insert into client

values (53, 'Nume', 'Prenume', '8798384946', null, null);

```

insert into client
values (53, 'Nume', 'Prenume', '8798384946', null, null);

```

Script Output x Query Result x

Task completed in 0.065 seconds

Error starting at line : 711 in command -
insert into client
values (53, 'Nume', 'Prenume', '8798384946', null, null)
Error report -
ORA-20002: Numarul de telefon pe care doriti sa il inserati nu este valid! Introduceti un numar de telefon mobil care incepe cu 07!
ORA-06512: at "ANDREEA_SORA1.EX11", line 36
ORA-04088: error during execution of trigger 'ANDREEA_SORA1.EX11'

Pentru rezolvarea cerinței am definit un pachet cu câte un contor care reține numărul de linii modificate/șterse din fiecare tabel utilizat. `Sql%ROWCOUNT` este o variabilă de sistem care este utilizată pentru a returna numărul de linii care sunt afectate de ultima instrucțiune executată.

12. Definiți un *trigger* de tip LDD. Declanșați *trigger*-ul.

Cerință:

Definiți un trigger LDD care permite modificări asupra bazei de date doar de către utilizatorul `andreea_sora1`, în intervalul orar 8-21. Toate modificările (cât și cele încercate în condiții nepermise) făcute asupra schemei se vor salva într-un tabel. Declanșați trigger-ul.

```

CREATE TABLE istoric_LDD (
utilizator VARCHAR2(40),
nume_baza_de_date VARCHAR2(40),
eveniment VARCHAR2(40),
nume_obiect VARCHAR2(40),
data_modificarii DATE,
eroare VARCHAR2(50)
);

```

```

CREATE OR REPLACE TRIGGER ex12
BEFORE CREATE OR ALTER OR DROP ON SCHEMA
DECLARE
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN

```

```

IF USER != UPPER('andreea_sora1') THEN
    INSERT INTO istoric_LDD
    VALUES(SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT,
SYS.DICTIONARY_OBJ_NAME, SYSDATE, 'alt user');
    commit;
    RAISE_APPLICATION_ERROR(-20900, 'Doar andreea_sora1 are permisiunea de
a modifica schema!');
ELSIF TO_CHAR(SYSDATE, 'HH24') NOT BETWEEN 8 AND 21 THEN
    INSERT INTO istoric_LDD
    VALUES(SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT,
SYS.DICTIONARY_OBJ_NAME, SYSDATE, 'in afara orelor permise');
    commit;
    RAISE_APPLICATION_ERROR(-20800, 'Schema nu poate fi actualizata decat
intre orele 8:00 si 21:00!');
ELSE
    INSERT INTO istoric_LDD
    VALUES(SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT,
SYS.DICTIONARY_OBJ_NAME, SYSDATE, 'succes');
    commit;
END IF;
END ex12;

```

SQL Worksheet | History

Worksheet | Query Builder

--12. Definiti un trigger LDD care permite modificari asupra bazei de date doar de catre utilizatorul andreea_sora1, in intervalul orar 8-21.
--Toate modificarile (cât și cele încercate în condiții nepermise) facute asupra schemei se vor salva într-un tabel. Declanșați trigger-ul.

```

CREATE TABLE istoric_LDD (
    utilizator VARCHAR2(40),
    nume_baza_de_date VARCHAR2(40),
    eveniment VARCHAR2(40),
    nume_obiect VARCHAR2(40),
    data_modificarii DATE,
    eroare VARCHAR2(50)
);

CREATE OR REPLACE TRIGGER ex12
BEFORE CREATE OR ALTER OR DROP ON SCHEMA
DECLARE
    PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
    IF USER != UPPER('andreea_sora1') THEN
        INSERT INTO istoric_LDD
        VALUES(SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME, SYSDATE, 'alt user');
        commit;
        RAISE_APPLICATION_ERROR(-20900, 'Doar andreea_sora1 are permisiunea de a modifica schema!');
    ELSIF TO_CHAR(SYSDATE, 'HH24') NOT BETWEEN 8 AND 21 THEN
        INSERT INTO istoric_LDD
        VALUES(SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME, SYSDATE, 'in afara orelor permise');
        commit;
        RAISE_APPLICATION_ERROR(-20800, 'Schema nu poate fi actualizata decat intre orele 8:00 si 21:00!');
    ELSE
        INSERT INTO istoric_LDD
        VALUES(SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME, SYSDATE, 'succes');
        commit;
    END IF;
END;

```

Script Output x

Task completed in 0.071 seconds

Trigger EX12 compiled

- Exemple modificări acceptate:

```
create table tabel_proba(
nume VARCHAR2(5)
);
```

```
drop table tabel_proba;
```

```
create table tabel_proba(
nume VARCHAR2(5)
);
```

```
alter table tabel_proba
add numar number;
```

```
select * from istoric_ldd;
```

```
create table tabel_proba(
nume VARCHAR2(5)
);

drop table tabel_proba;

create table tabel_proba(
nume VARCHAR2(5)
);

alter table tabel_proba
add numar number;

select * from istoric_ldd;
```

Script Output x Query Result x						
SQL All Rows Fetched: 4 in 0.001 seconds						
UTILIZATOR	NUME_BAZA_DE_DATE	EVENIMENT	NUME_OBIECT	DATA_MODIFICARII	EROARE	
1 ANDREEA_SORA1	XE	CREATE	TABEL_PROBA	26-DEC-21	succes	
2 ANDREEA_SORA1	XE	CREATE	TABEL_PROBA	26-DEC-21	succes	
3 ANDREEA_SORA1	XE	DROP	TABEL_PROBA	26-DEC-21	succes	
4 ANDREEA_SORA1	XE	ALTER	TABEL_PROBA	26-DEC-21	succes	

- Declanșare trigger pentru excepția modificării schemei în afara intervalului orar permis:

```
create table tabel(
cod NUMBER
);
```

```

select * from istoric_ldd;

create table tabel(
cod NUMBER
);

```

Script Output x | Task completed in 0.067 seconds

```

create table tabel(
cod NUMBER
)
Error report -
ORA-00604: error occurred at recursive SQL level 1
ORA-20800: Schema nu poate fi actualizata decat intre orele 8:00 si 21:00!
ORA-06512: at line 13
00604. 00000 - "error occurred at recursive SQL level %s"
*Cause:      An error occurred while processing a recursive SQL statement
              (a statement applying to internal dictionary tables).
*Action:     If the situation described in the next error on the stack
              can be corrected, do so; otherwise contact Oracle Support.

```

Totuși, în acest caz, inserarea se va realiza oricum în istoricul nostru:

```

select * from istoric_ldd;

create table tabel(
cod NUMBER
);

```

Script Output x | Query Result x | All Rows Fetched: 5 in 0.001 seconds

UTILIZATOR	NUME_BAZA_DE_DATA	EVENIMENT	NUME_OBIECT	DATA_MODIFICARII	EROARE
1 ANDREEA_SORA1 XE		CREATE	TABEL_PROBA	26-DEC-21	succes
2 ANDREEA_SORA1 XE		CREATE	TABEL_PROBA	26-DEC-21	succes
3 ANDREEA_SORA1 XE		DROP	TABEL_PROBA	26-DEC-21	succes
4 ANDREEA_SORA1 XE		ALTER	TABEL_PROBA	26-DEC-21	succes
5 ANDREEA_SORA1 XE		CREATE	TABEL	26-DEC-21	in afara orelor permise

Pentru a putea face insert-ul și pe raise și a putea folosi commit, am utilizat un trigger autonom. Spre deosebire de triggerile obișnuite, triggerile autonome pot conține instrucțiuni de control al tranzacțiilor, cum ar fi COMMIT sau ROLLBACK. Pragma AUTONOMOUS_TRANSACTION schimbă modul în care funcționează un subprogram în cadrul unei tranzacții. Acest cuvânt cheie va instrui compilatorul să trateze această tranzacție ca o tranzacție separată, iar salvarea/eliminarea în interiorul acestui bloc nu se va reflecta în tranzacția principală. Folosirea COMMIT sau ROLLBACK este obligatorie înainte de a ieși din această tranzacție autonomă la tranzacția principală deoarece în orice moment poate fi activă o singură tranzacție. (lucru demonstrat în prima temă de curs) Pragma-urile sunt procesate în timpul compilării, nu în timpul rulării.

13. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.

```
CREATE OR REPLACE PACKAGE ex13 IS
  PROCEDURE seteaza_valoarea_comenzii;
  PROCEDURE lista_comenzi;
  FUNCTION top3(ume1 client.ume_client%TYPE, ume2
client.ume_client%TYPE, ume3 client.ume_client%TYPE, ume_res
restaurant.ume_restaurant%TYPE,
  data_com comanda.data_comanda%TYPE) RETURN VARCHAR2;
  PROCEDURE detalii(ume client.ume_client%TYPE, optiune NUMBER);
END ex13;
```

```
CREATE OR REPLACE PACKAGE BODY ex13 IS
```

```
  PROCEDURE seteaza_valoarea_comenzii IS
    TYPE tab_imb IS TABLE OF NUMBER;
    TYPE comanda_tab IS TABLE OF comanda.cod_comanda%TYPE INDEX BY
    BINARY_INTEGER;
    v_coduri_preparate tab_imb := tab_imb();
    tabel_comenzi comanda_tab;
    v_valoare comanda.valoare%type;
    nr NUMBER;
    valoare_curenta comanda.valoare%type;
  BEGIN
    select cod_comanda
    bulk collect into tabel_comenzi
    from comanda;

    FOR i IN tabel_comenzi.FIRST..tabel_comenzi.LAST LOOP
      v_valoare := 0;

      select count(cod_preparat)
      into nr
      from cuprinde c
      where c.cod_comanda = tabel_comenzi(i);

      FOR j IN 1..nr LOOP
        v_coduri_preparate.EXTEND;
      END LOOP;

      select cod_preparat
```

```

    bulk collect into v_coduri_preparate
    from cuprinde c
    where c.cod_comanda= tabel_comenzi(i);

    FOR k IN v_coduri_preparate.FIRST..v_coduri_preparate.LAST LOOP
        select p.pret*c.cantitate
        into valoare_curenta
        from preparat p, cuprinde c
        where p.cod_preparat = v_coduri_preparate(k) and c.cod_preparat =
v_coduri_preparate(k) and c.cod_comanda = tabel_comenzi(i);

        v_valoare := v_valoare + valoare_curenta;
    END LOOP;

    update comanda
    set valoare = v_valoare
    where cod_comanda = tabel_comenzi(i);

    v_coduri_preparate.DELETE;
    dbms_output.put_line('Comanda ' || tabel_comenzi(i) || ' are valoarea de ' ||
v_valoare || ' lei.');
```

```

    dbms_output.put_line('Update realizat cu succes!');
    dbms_output.new_line();
END LOOP;
END seteaza_valoarea_comenzii;

PROCEDURE lista_comenzi IS
    TYPE comanda_record IS RECORD
        (cod_comanda.cod_comanda%TYPE,
        data_plasare comanda.data_comanda%TYPE,
        plata comanda.tip_plata%TYPE);
    TYPE tablou_comanda IS TABLE OF comanda_record INDEX BY
    BINARY_INTEGER;
    TYPE refcursor IS REF CURSOR;
    CURSOR c IS
        select nume_preparat, CURSOR (select c.cod_comanda, c.data_comanda, c.tip_plata
        from comanda c, cuprinde cu
        where c.cod_comanda = cu.cod_comanda and cu.cod_preparat =
p.cod_preparat)
        from preparat p;
    v_cursor refcursor;
    v_comanda tablou_comanda;
```

```

v_preparat preparat.nume_preparat%TYPE;
nr NUMBER;
zi NUMBER;
an NUMBER;
luna_cu_spatii VARCHAR2(15);
luna_fara_spatii VARCHAR2(15);
BEGIN
OPEN c;
LOOP
FETCH c INTO v_preparat, v_cursor;
EXIT WHEN c%NOTFOUND;
dbms_output.put_line('Preparatul: ' || v_preparat);
nr := 0;
LOOP
FETCH v_cursor BULK COLLECT INTO v_comanda;
nr := v_comanda.COUNT;
IF nr = 0 THEN
dbms_output.put_line('Nu exista comenzi care sa cuprinda preparatul
respectiv!');
ELSE
dbms_output.put_line('Comenzile care cuprind preparatul ' || v_preparat || ':');
FOR i IN v_comanda.FIRST..v_comanda.LAST LOOP
zi := extract (day from v_comanda(i).data_plasare);

select to_char(v_comanda(i).data_plasare, 'Month',
'NLS_DATE_LANGUAGE = Romanian')
into luna_cu_spatii
from dual;

luna_fara_spatii := RTRIM(luna_cu_spatii, ' ');
an := extract(year from v_comanda(i).data_plasare);
dbms_output.put_line('Comanda cu codul ' || v_comanda(i).cod || ', plasata in
data de ' || zi || ' ' || luna_fara_spatii || ' ' || an
|| ' si care are tipul platii ' || v_comanda(i).plata || '.');
END LOOP;
END IF;
EXIT WHEN v_cursor%NOTFOUND;
END LOOP;
dbms_output.new_line;
END LOOP;
CLOSE c;
END lista_comenzi;

```



```

FUNCTION top3(ume1 client.ume_client%TYPE, ume2
client.ume_client%TYPE, ume3 client.ume_client%TYPE, ume_res
restaurant.ume_restaurant%TYPE,
    data_com comanda.data_comanda%TYPE)
    RETURN VARCHAR2 IS
v_cod_res NUMBER;
v_cod_cl1 client.cod_client%TYPE;
v_cod_cl2 client.cod_client%TYPE;
v_cod_cl3 client.cod_client%TYPE;
nr_cl1 NUMBER;
nr_cl2 NUMBER;
nr_cl3 NUMBER;
aux NUMBER;
aux_ume client.ume_client%TYPE;
ume1_copie client.ume_client%TYPE;
ume2_copie client.ume_client%TYPE;
ume3_copie client.ume_client%TYPE;
mesaj VARCHAR2(200);
BEGIN
    select cod_client
    into v_cod_cl1
    from client
    where upper(ume_client) = upper(ume1);

    select cod_client
    into v_cod_cl2
    from client
    where upper(ume_client) = upper(ume2);

    select cod_client
    into v_cod_cl3
    from client
    where upper(ume_client) = upper(ume3);

    select cod_restaurant
    into v_cod_res
    from restaurant
    where upper(ume_restaurant) = upper(ume_res);

    select count(*)
    into nr_cl1

```

```

from istoric_comanda i, client c, comanda co, restaurant r
where upper(c.ume_client) = upper(ume1) and c.cod_client = i.cod_client and
i.cod_comanda = co.cod_comanda and co.cod_restaurant = r.cod_restaurant
and upper(r.ume_restaurant) = upper(ume_res) and co.data_comanda > data_com;

```

```

select count(*)
into nr_cl2
from istoric_comanda i, client c, comanda co, restaurant r
where upper(c.ume_client) = upper(ume2) and c.cod_client = i.cod_client and
i.cod_comanda = co.cod_comanda and co.cod_restaurant = r.cod_restaurant
and upper(r.ume_restaurant) = upper(ume_res) and co.data_comanda > data_com;

```

```

select count(*)
into nr_cl3
from istoric_comanda i, client c, comanda co, restaurant r
where upper(c.ume_client) = upper(ume3) and c.cod_client = i.cod_client and
i.cod_comanda = co.cod_comanda and co.cod_restaurant = r.cod_restaurant
and upper(r.ume_restaurant) = upper(ume_res) and co.data_comanda > data_com;

```

```

ume1_copie := ume1;
ume2_copie := ume2;
ume3_copie := ume3;

```

```

IF (nr_cl1 < nr_cl2) THEN
    aux := nr_cl1;
    nr_cl1 := nr_cl2;
    nr_cl2 := aux;
    aux_ume := ume1_copie;
    ume1_copie := ume2_copie;
    ume2_copie := aux_ume;
END IF;

```

```

IF (nr_cl1 < nr_cl3) THEN
    aux := nr_cl1;
    nr_cl1 := nr_cl3;
    nr_cl3 := aux;
    aux_ume := ume1_copie;
    ume1_copie := ume3_copie;
    ume3_copie := aux_ume;
END IF;

```

```

IF (nr_cl2 < nr_cl3) THEN

```

```

    aux := nr_cl2;
    nr_cl2 := nr_cl3;
    nr_cl3 := aux;
    aux_nume := nume2_copie;
    nume2_copie := nume3_copie;
    nume3_copie := aux_nume;
END IF;

```

```

    mesaj := 'Top 3 dupa numarul de comenzi plasate in restaurantul ' || initcap(num_res)
|| ' dupa data de ' || data_com || ': ' || nr_cl1
|| ' - ' || initcap(nume1_copie) || '; ' || nr_cl2 || ' - ' || initcap(nume2_copie) || '; ' || nr_cl3 || '
- ' || initcap(nume3_copie);
RETURN mesaj;

```

EXCEPTION

WHEN NO_DATA_FOUND THEN

IF v_cod_cl1 IS NULL THEN

 mesaj := 'Nu exista client cu numele ' || initcap(nume1) || '!';

ELSIF v_cod_cl2 IS NULL THEN

 mesaj := 'Nu exista client cu numele ' || initcap(nume2) || '!';

ELSIF v_cod_cl3 IS NULL THEN

 mesaj := 'Nu exista client cu numele ' || initcap(nume3) || '!';

ELSE

 mesaj := 'Nu exista restaurant cu numele ' || initcap(num_res) || '!';

END IF;

RETURN mesaj;

WHEN TOO_MANY_ROWS THEN

IF v_cod_cl1 IS NULL THEN

 mesaj := 'Exista mai multi clienti cu numele ' || initcap(nume1) || '!';

ELSIF v_cod_cl2 IS NULL THEN

 mesaj := 'Exista mai multi clienti cu numele ' || initcap(nume2) || '!';

ELSIF v_cod_cl3 IS NULL THEN

 mesaj := 'Exista mai multi clienti cu numele ' || initcap(nume3) || '!';

ELSE

 mesaj := 'Exista mai multe restaurante cu numele ' || initcap(num_res) || '!';

END IF;

RETURN mesaj;

WHEN OTHERS THEN

 RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');

END top3;

PROCEDURE detalii(num_client.num_client%TYPE, optiune NUMBER) IS

TYPE informatii IS RECORD

```
(nr_persoane rezervare.nr_persoane%TYPE,
 data_r rezervare.data_rezervare%TYPE,
 data_e eveniment.data_eveniment%TYPE,
 nume_res restaurant.nume_restaurant%TYPE,
 tip_e tip_eveniment.descriere%TYPE);
v_inf informatii;
```

CURSOR c(parametru client.nume_client%type) IS

```
select r.nr_persoane, r.data_rezervare , e.data_eveniment, res.nume_restaurant,
t.descriere
from client c,REZERVARE r, EVENIMENT e, RESTAURANT res, FACE f,
tip_eveniment t
where f.cod_rezervare=r.cod_rezervare and f.cod_restaurant=res.cod_restaurant
and f.cod_eveniment=e.cod_eveniment and e.cod_tip = t.cod_tip and f.cod_client =
c.cod_client and upper(c.nume_client) = upper(parametru);
```

TYPE comanda_tip IS REF CURSOR RETURN comanda%ROWTYPE;

v_comanda comanda_tip;

v_com comanda%ROWTYPE;

contor NUMBER :=1;

ok NUMBER := 0;

TYPE clienti_tab IS TABLE OF istoric_comanda.cod_client%TYPE INDEX BY
BINARY_INTEGER;

coduri_clienti clienti_tab;

v_cod_client client.cod_client%TYPE;

BEGIN

select cod_client

into v_cod_client

from client

where upper(nume_client) = upper(nume);

IF optiune = 1 THEN

dbms_output.put_line('Rezervari pentru clientul ' || initcap(nume) || ':');

OPEN c(nume);

LOOP

FETCH c INTO v_inf;

EXIT WHEN c%NOTFOUND;

```
dbms_output.put_line(contor || '. Numar persoane: ' || v_inf.nr_persoane || ', Data
rezervare: ' || v_inf.data_r || ', Data eveniment: ' || v_inf.data_e
|| ', Nume restaurant: ' || v_inf.nume_res || ', Tip eveniment: ' || v_inf.tip_e);
contor := contor + 1;
```

```

END LOOP;
CLOSE c;
ELSIF optiune = 2 THEN
    dbms_output.put_line('Comenzi pentru clientul ' || initcap(ume) || ':');
    OPEN v_comanda FOR select co.*
        from comanda co, client c, istoric_comanda i
        where upper(c.ume_client) = upper(ume) and c.cod_client =
i.cod_client and i.cod_comanda = co.cod_comanda;
    LOOP
        FETCH v_comanda INTO v_com;
        EXIT WHEN v_comanda%NOTFOUND;
        dbms_output.put_line(contor || '. Comanda ' || v_com.cod_comanda || ' cu valoarea
de ' || v_com.valoare || ' lei, cu tipul platii ' || v_com.tip_plata);
        contor := contor + 1;
    END LOOP;
    CLOSE v_comanda;
ELSIF optiune = 3 THEN
    select distinct(a.cod_client)
    bulk collect into coduri_clienti
    from istoric_comanda a
    where not exists (select 1
        from COMANDA c
        where mod(c.valoare, 10) = 0 and not exists (select 1
            from istoric_comanda b
            where c.cod_comanda=b.cod_comanda and
b.cod_client=a.cod_client));
    FOR i IN coduri_clienti.FIRST..coduri_clienti.LAST LOOP
        IF coduri_clienti(i) = v_cod_client THEN
            ok := 1;
        END IF;
    END LOOP;
    IF ok = 1 THEN
        dbms_output.put_line('Pentru ' || initcap(ume) || ' nu exista comenzi cu valoarea
un numar divizibil cu 10 pe care sa nu le fi facut.');
```

```

    ELSE
        dbms_output.put_line('Pentru ' || initcap(ume) || ' exista comenzi cu valoarea un
numar divizibil cu 10 pe care sa nu le fi facut.');
```

```

    END IF;
ELSE
    dbms_output.put_line('Optiune invalida!');
END IF;

```

```

EXCEPTION
WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20000,'Nu exista clientul cu numele transmis ca
parametru!');
WHEN TOO_MANY_ROWS THEN
    RAISE_APPLICATION_ERROR(-20001,'Exista mai multi clienti cu numele
transmis ca parametru!');
WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END detalii;
END ex13;

```

SQL Worksheet History

Worksheet Query Builder

```

--13. Definiti un pachet care sa contina toate obiectele definite in cadrul proiectului.
CREATE OR REPLACE PACKAGE ex13 IS
    PROCEDURE seteaza_valoarea_comenzii;
    PROCEDURE lista_comenzi;
    FUNCTION top3(numel client.nume_client%TYPE, nume2 client.nume_client%TYPE, nume3 client.nume_client%TYPE, nume_res restaurant.nume_restaurant%TYPE,
        data_com comanda.data_comanda%TYPE) RETURN VARCHAR2;
    PROCEDURE detalii(nume client.nume_client%TYPE, optiune NUMBER);
END ex13;

CREATE OR REPLACE PACKAGE BODY ex13 IS

    PROCEDURE seteaza_valoarea_comenzii IS
        TYPE tab_imb IS TABLE OF NUMBER;
        TYPE comanda_tab IS TABLE OF comanda.cod_comanda%TYPE INDEX BY BINARY_INTEGER;
        v_coduri_preparate tab_imb := tab_imb();
        tabel_comenzi comanda_tab;
        v_valoare comanda.valoare%type;
        nr NUMBER;
        valoare_curenta comanda.valoare%type;
    BEGIN
        select cod_comanda
        bulk collect into tabel_comenzi
        from comanda;

        FOR i IN tabel_comenzi.FIRST..tabel_comenzi.LAST LOOP
            v_valoare := 0;

            select count(cod_preparat)
            into nr

```

Script Output x

Task completed in 0.192 seconds

Package EX13 compiled

Package Body EX13 compiled

Exemplu apel din pachet al procedurii de la exercițiul 9:

execute ex13.detalii('frusinoiu', 2);

```

execute ex13.detalii('frusinoiu', 2);

```

Script Output x

Task completed in 0.063 seconds

Package Body EX13 compiled

Comenzi pentru clientul Frusinoiu:

1. Comanda 501 cu valoarea de 47 lei, cu tipul platii cash

PL/SQL procedure successfully completed.

14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).

Cerință:

Definiți un pachet care să permită gestiunea angajaților (anumite modificări, afișarea unor anumite detalii etc). Pentru aceasta, se vor defini următoarele:

- Un record în care se vor înregistra numele, prenumele și salariul unui angajat;
- O procedură care afișează numele, prenumele și salariul unui angajat cu un cod transmis ca parametru;
- Un cursor care reține o listă a angajaților pe un anumit job, ordonați descrescător după salariu;
- O procedură care afișează lista angajaților în ordine descrescătoare după salariu pentru un anumit job al cărui cod este transmis ca parametru;
- O funcție care returnează codul adresei al cărui oraș este transmis ca parametru;
- O procedură care mărește salariile cu 5% pentru toți angajații conduși direct sau indirect de un manager transmis ca parametru, dintr-un restaurant care are adresa corespunzătoare codului adresei transmis de asemenea ca parametru;
- Un tablou indexat ce reține codurile angajaților din istoricul job-urilor;
- O funcție care returnează numărul angajaților care au avut 2 job-uri de-a lungul timpului;
- O funcție care returnează numărul angajaților dintr-un restaurant al cărui cod este transmis ca parametru;
- O procedură care modifică salariul unui angajat al cărui cod este transmis ca parametru. Salariul nou se va transmite și el ca parametru. Acest salariu trebuie să corespundă limitelor salariale pe job-ul respective;
- O procedură care marchează faptul că un angajat a fost concediat în ziua respectivă.

Exemplificați funcționalitățile pachetului prin apeluri corespunzătoare.

CREATE OR REPLACE PACKAGE ex14 IS

TYPE ang_record IS RECORD(

nume angajat.nume_angajat%TYPE,
prenume angajat.prenume_angajat%TYPE,
salariu_maxim angajat.salariu%TYPE);

PROCEDURE detalii_angajat (cod angajat.cod_angajat%TYPE);

CURSOR lista_ang(v_job job.cod_job%TYPE) IS

select *

```

from angajat
where cod_job = v_job
order by salariu desc;
PROCEDURE lista_angajati(cod_job.cod_job%TYPE);
FUNCTION codul_adresei(v_oras adresa.oras%TYPE) RETURN
adresa.cod_adresa%TYPE;
PROCEDURE mareste_salariul(cod_angajat.cod_angajat%TYPE, adresa_cod
adresa.cod_adresa%TYPE);
TYPE angajati IS TABLE OF istoric_job.cod_angajat%TYPE INDEX BY
BINARY_INTEGER;
FUNCTION nr_ang_cu_2_joburi RETURN NUMBER;
FUNCTION nr_ang_in_res(cod_restaurant.cod_restaurant%TYPE) RETURN
NUMBER;
PROCEDURE update_salariu(cod_angajat.cod_angajat%TYPE, salariu_nou
angajat.salariu%TYPE);
PROCEDURE concediaza(cod_istoric_job.cod_angajat%TYPE);
END ex14;

```

CREATE OR REPLACE PACKAGE BODY ex14 IS

```

PROCEDURE detalii_angajat(cod_angajat.cod_angajat%TYPE) IS
    v_cod angajat.cod_job%TYPE;
    v_angajat ang_record;
BEGIN
    select cod_angajat
    into v_cod
    from angajat
    where cod_angajat = cod;

    select nume_angajat, prenume_angajat, salariu
    into v_angajat.nume, v_angajat.prenume, v_angajat.salariu_maxim
    from angajat
    where cod_angajat = v_cod;

    dbms_output.put_line('Angajatul ' || v_angajat.nume || ' ' || v_angajat.prenume || ' are
salariul ' || v_angajat.salariu_maxim || ' lei.');
```

EXCEPTION

```

    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000,'Nu exista angajat cu codul transmis ca
parametru!');
    WHEN OTHERS THEN

```



```
        RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END detalii_angajat;

PROCEDURE lista_angajati(cod job.cod_job%TYPE) IS
    nume job.nume_job%TYPE;
    contor NUMBER := 1;
BEGIN
    select nume_job
    into nume
    from job
    where cod_job = cod;

    dbms_output.put_line('Angajatii cu jobul ' || nume || ', ordonati descrescator dupa
salariu:');

    FOR i IN lista_ang(cod) LOOP
        dbms_output.put_line(contor || '. Angajatul ' || i.nume_angajat || ' ' ||
i.prenume_angajat || ', cu salariul ' || i.salariu || '.');
        contor := contor + 1;
    END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000,'Nu exista job cu codul transmis ca
parametru!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END lista_angajati;

FUNCTION codul_adresei(v_oras adresa.oras%TYPE)
RETURN adresa.cod_adresa%TYPE IS
    v_cod adresa.cod_adresa%TYPE;
BEGIN
    select cod_adresa
    into v_cod
    from adresa
    where upper(oras) = upper(v_oras);

    RETURN v_cod;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000,'Nu exista adresa in orasul transmis ca
parametru!');
```

```
WHEN TOO_MANY_ROWS THEN
    RAISE_APPLICATION_ERROR(-20001,'Exista mai multe adrese in orasul cu
numele transmis ca parametru!');
WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END codul_adresei;
```

```
PROCEDURE mareste_salariul(cod_angajat%TYPE, adresa_cod
adresa.cod_adresa%TYPE) IS
```

```
    v_cod_restaurant restaurant.cod_restaurant%TYPE;
    v_sef angajat.cod_sef%TYPE;
```

```
BEGIN
```

```
    select cod_restaurant
    into v_cod_restaurant
    from restaurant
    where cod_adresa = adresa_cod;
```

```
    select cod_angajat
    into v_sef
    from angajat
    where cod_angajat = cod and cod_sef is null;
```

```
    update angajat
    set salariu = salariu * 0.05 + salariu
    where cod_sef in (select cod_angajat
                      from angajat
                      start with cod_angajat = cod
                      connect by prior cod_angajat = cod_sef);
```

```
    dbms_output.put_line('Update-uri realizate cu succes!');
```

```
EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000,'Nu exista manager cu codul transmis
ca parametru!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END mareste_salariul;
```

```
FUNCTION nr_ang_cu_2_joburi
RETURN NUMBER IS
    v_ang angajati;
```

```

cod istoric_job.cod_angajat%TYPE;
contor NUMBER := 0;
BEGIN
    select cod_angajat
    bulk collect into v_ang
    from istoric_job
    where data_final is not null;

    FOR i IN v_ang.FIRST..v_ang.LAST LOOP
        select cod_angajat
        into cod
        from istoric_job
        where data_final is null and cod_angajat = v_ang(i);

        IF cod = v_ang(i) THEN
            contor := contor + 1;
        END IF;
    END LOOP;
    RETURN contor;
END nr_ang_cu_2_joburi;

FUNCTION nr_ang_in_res(cod restaurant.cod_restaurant%TYPE)
RETURN NUMBER IS
    cod_r restaurant.cod_restaurant%TYPE;
    nr NUMBER;
BEGIN
    select cod_restaurant
    into cod_r
    from restaurant
    where cod_restaurant = cod;

    select count(*)
    into nr
    from angajat
    where cod_restaurant = cod_r;

    RETURN nr;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000,'Nu exista restaurant cu codul transmis
ca parametru!');
    WHEN OTHERS THEN

```

```
        RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
    END nr_ang_in_res;

    PROCEDURE update_salariu(cod angajat.cod_angajat%TYPE, salariu_nou
angajat.salariu%TYPE) IS
        v_cod angajat.cod_angajat%TYPE;
        v_cod_job angajat.cod_job%TYPE;
        limita_inf job.salariu_minim%TYPE;
        limita_sup job.salariu_maxim%TYPE;
    BEGIN
        select cod_angajat
        into v_cod
        from angajat
        where cod_angajat = cod;

        select cod_job
        into v_cod_job
        from angajat
        where cod_angajat = v_cod;

        select salariu_minim, salariu_maxim
        into limita_inf, limita_sup
        from job
        where cod_job = v_cod_job;

        IF (limita_inf is null and limita_sup is null) or (limita_inf is null and salariu_nou <=
limita_sup) or
        (salariu_nou >= limita_inf and limita_sup is null) or (salariu_nou >= limita_inf and
salariu_nou <= limita_sup) THEN
            update angajat
            set salariu = salariu_nou
            where cod_angajat = v_cod;

            dbms_output.put_line('Update realizat cu succes!');
        ELSE
            dbms_output.put_line('Salariul nou introdus nu este conform limitelor salariale!
Limita inferioara: '
            || limita_inf || '; Limita superioara: ' || limita_sup);
        END IF;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
```

```
        RAISE_APPLICATION_ERROR(-20000,'Nu exista angajat cu codul transmis ca
parametru!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END update_salariu;

PROCEDURE concediaza(cod_istoric_job.cod_angajat%TYPE) IS
    v_cod_istoric_job.cod_angajat%TYPE;
    nr NUMBER;
BEGIN
    select cod_angajat
    into v_cod
    from istoric_job
    where cod_angajat = cod;

    update istoric_job
    set data_final = sysdate
    where cod_angajat = v_cod and data_final is null;

    nr := sql%rowcount;
    IF nr = 1 THEN
        dbms_output.put_line('Angajatul ' || v_cod || ' a fost concediat!');
    ELSE
        dbms_output.put_line('Angajatul ' || v_cod || ' a fost deja concediat!');
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000,'Nu exista angajat cu codul transmis ca
parametru!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002,'Alta eroare!');
END concediaza;

END ex14;
```

```

--14. Definiti un pachet care sa permita gestiunea angajatilor (anumite modificari, afisarea unor anumite detalii etc). Pentru aceasta, se vor defini urmatoarele:
-- Un record in care se vor inregistra numele, prenumele si salariul unui angajat;
-- O procedura care afiseaza numele, prenumele si salariul unui angajat cu un cod transmis ca parametru;
-- Un cursor care retine o lista a angajatilor pe un anumit job, ordonati descrescator dupa salariu;
-- O procedura care afiseaza lista angajatilor in ordine descrescatoare dupa salariu pentru un anumit job al carui cod este transmis ca parametru;
-- O functie care returneaza codul adresei al carui oras este transmis ca parametru;
-- O procedura care maresta salariile cu 5% pentru toti angajatii condusi direct
--sau indirect de un manager transmis ca parametru, dintr-un restaurant care are adresa corespunzatoare codului adresei transmis de asemenea ca parametru;
-- Un tablou indexat ce retine codurile angajatilor din istoricul job-urilor;
-- O functie care returneaza numarul angajatilor care au avut 2 job-uri de-a lungul timpului;
-- O functie care returneaza numarul angajatilor dintr-un restaurant al carui cod este transmis ca parametru;
-- O procedura care modifica salariul unui angajat al carui cod este transmis ca
--parametru. Salariul nou se va transmite si el ca parametru. Acest salariu trebuie sa corespunda limitelor salariale pe job-ul respectiv;
-- Exemplificati functionalitatile pachetului prin apeluri corespunzatoare.

CREATE OR REPLACE PACKAGE ex14 IS
  TYPE ang_record IS RECORD(
    nume angajat.nume_angajat%TYPE,
    prenume angajat.prenume_angajat%TYPE,
    salariu_maxim angajat.salariu%TYPE);
  PROCEDURE detalii_angajat (cod angajat.cod_angajat%TYPE);
  CURSOR lista_ang(v_job job.cod_job%TYPE) IS
    select *
    from angajat
    where cod_job = v_job
    order by salariu desc;
  PROCEDURE lista_angajati(cod job.cod_job%TYPE);
  FUNCTION codul_adresei(v_oras adresa.oras%TYPE) RETURN adresa.cod_adresa%TYPE;
  PROCEDURE maresta_salariul(cod angajat.cod_angajat%TYPE, adresa cod_adresa.cod_adresa%TYPE);
END;

```

Package EX14 compiled

Package Body EX14 compiled

- Apeluri procedura detalii_angajat:

execute ex14.detalii_angajat(30);

```

execute ex14.detalii_angajat(30);

```

Script Output x

Task completed in 0.042 seconds

```

PL/SQL procedure successfully completed.

Angajatul Ion Mihail are salariul 9900 lei.

PL/SQL procedure successfully completed.

```

execute ex14.detalii_angajat(70);

```

execute ex14.detalii_angajat(70);

```

Script Output x

Task completed in 0.081 seconds

```

Error starting at line : 1,343 in command -
BEGIN ex14.detalii_angajat(70); END;
Error report -
ORA-20000: Nu exista angajat cu codul transmis ca parametru!
ORA-06512: at "ANDREEA_SORA1.EX14", line 21
ORA-06512: at line 1
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.

```

- Apeluri procedura lista_angajati:

execute ex14.lista_angajati(1000);

```
execute ex14.lista_angajati(1000);
```

Script Output x

Task completed in 0.04 seconds

Angajatii cu jobul manager, ordonati descrescator dupa salariu:

1. Angajatul Popescu Ion, cu salariul 25000.
2. Angajatul Dragomirescu Laur, cu salariul 22995.
3. Angajatul Pop Vlad, cu salariul 21000.
4. Angajatul Sora George, cu salariul 20900.
5. Angajatul Ifrim George, cu salariul 20300.
6. Angajatul Petrescu Laurentiu, cu salariul 18900.
7. Angajatul Bojici Matei, cu salariul 18000.

PL/SQL procedure successfully completed.

execute ex14.lista_angajati(1090);

```
execute ex14.lista_angajati(1090);
```

Script Output x

Task completed in 0.039 seconds

Error starting at line : 1,343 in command -
 BEGIN ex14.lista_angajati(1090); END;
 Error report -
 ORA-20000: Nu exista job cu codul transmis ca parametru!
 ORA-06512: at "ANDREEA_SORA1.EX14", line 43
 ORA-06512: at line 1
 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.

- Apeluri funcție coduri_adrese:

BEGIN

dbms_output.put_line('Codul adresei cu orasul Sinaia: ' || ex14.codul_adresei('Sinaia'));

END;

```

BEGIN
    dbms_output.put_line('Codul adresei cu orasul Sinaia: ' || ex14.codul_adresei('Sinaia'));
END;

```

Script Output x

Task completed in 0.041 seconds

PL/SQL procedure successfully completed.

Codul adresei cu orasul Sinaia: 10

PL/SQL procedure successfully completed.

BEGIN

```

    dbms_output.put_line('Codul adresei cu orasul Ploiesti: ' ||
ex14.codul_adresei('Ploiesti'));
END;

```

```

BEGIN
    dbms_output.put_line('Codul adresei cu orasul Ploiesti: ' || ex14.codul_adresei('Ploiesti'));
END;

```

Script Output x

Task completed in 0.088 seconds

BEGIN

```

    dbms_output.put_line('Codul adresei cu orasul Ploiesti: ' || ex14.codul_adresei('Ploiesti'));
END;

```

Error report -

ORA-20001: **Exista mai multe adrese in orasul cu numele transmis ca parametru!**

ORA-06512: at "ANDREEA_SORA1.EX14", line 62

ORA-06512: at line 2

BEGIN

```

    dbms_output.put_line('Codul adresei cu orasul Mizil: ' || ex14.codul_adresei('Mizil'));
END;

```

```

BEGIN
    dbms_output.put_line('Codul adresei cu orasul Mizil: ' || ex14.codul_adresei('Mizil'));
END;

```

Script Output x

Task completed in 0.084 seconds

END;

Error report -

ORA-20000: **Nu exista adresa in orasul transmis ca parametru!**

ORA-06512: at "ANDREEA_SORA1.EX14", line 60

ORA-06512: at line 2

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.

- Apeluri procedura mareste_salariul:

Această procedură a fost gândită să fie apelată împreună cu funcția `codul_adresei`, pentru ca utilizatorul să introducă de la tastatură codul unui manager și orașul în care lucrează, calculându-se codul adresei din orașul respectiv folosind funcția mai sus menționată.

DECLARE

```
v_oras adresa.oras%TYPE := '&oras';
cod_manager angajat.cod_angajat%TYPE := '&cod';
cod_ad adresa.cod_adresa%TYPE;
BEGIN
    cod_ad := ex14.codul_adresei(v_oras);
    ex14.mareste_salariul(cod_manager, cod_ad);
END;
```

- Pentru `oras = Busteni` și `cod manager = 10`:

```
DECLARE
    v_oras adresa.oras%TYPE := '&oras';
    cod_manager angajat.cod_angajat%TYPE := '&cod';
    cod_ad adresa.cod_adresa%TYPE;
BEGIN
    cod_ad := ex14.codul_adresei(v_oras);
    ex14.mareste_salariul(cod_manager, cod_ad);
END;
```

Script Output x

Task completed in 5.996 seconds

```
cod_ad adresa.cod_adresa%TYPE;
BEGIN
    cod_ad := ex14.codul_adresei(v_oras);
    ex14.mareste_salariul(cod_manager, cod_ad);
END;
new:DECLARE
v_oras adresa.oras%TYPE := 'busteni';
cod_manager angajat.cod_angajat%TYPE := '10';
cod_ad adresa.cod_adresa%TYPE;
BEGIN
    cod_ad := ex14.codul_adresei(v_oras);
    ex14.mareste_salariul(cod_manager, cod_ad);
END;
Update-uri realizate cu succes!
```

PL/SQL procedure successfully completed.

Înainte de update:

```
select a.*
from angajat a, restaurant r, adresa ad
where a.cod_restaurant = r.cod_restaurant and r.cod_adresa = ad.cod_adresa and ad.oras = 'Busteni';
```

Script Output x Query Result x

SQL | All Rows Fetched: 8 in 0.004 seconds

	COD_ANGAJAT	NUME_ANGAJAT	PRENUME_ANGAJAT	EMAIL	NR_TELEFON	SALARIU	DATA_ANGAJARE	COMISION	COD_SEF	COD_RESTAURANT	COD_JOB
1	15	Petrescu	Marius	email16@yahoo.com	0712345678	9000	12-JAN-21	0.35	11	101	1001
2	13	Neagu	Marian	email14@yahoo.com	0728647009	2500	12-MAR-21	0.1	11	101	1004
3	12	Ionescu	Ana	email13@yahoo.com	0738689839	3000	02-JAN-21	(null)	11	101	1002
4	11	Gheorghe	Mihai	email12@yahoo.com	0728622839	10400	10-DEC-20	0.1	10	101	1005
5	10	Popescu	Ion	email11@yahoo.com	0728647839	25000	02-OCT-20	(null)	(null)	101	1000
6	16	Lazar	Nicoleta	email17@yahoo.com	0711223344	9000	01-FEB-21	0.2	11	101	1001
7	17	Ion	Dan	email18@yahoo.com	0701234985	5000	18-APR-21	(null)	15	101	1006
8	14	Toma	Daniela	email15@yahoo.com	0728678901	1999	02-SEP-20	(null)	11	101	1003

După update:

```
select a.*
from angajat a, restaurant r, adresa ad
where a.cod_restaurant = r.cod_restaurant and r.cod_adresa = ad.cod_adresa and ad.oras = 'Busteni';
```

	COD_ANGAJAT	NUME_ANGAJAT	PRENUME_ANGAJAT	EMAIL	NR_TELEFON	SALARIU	DATA_ANGAJARE	COMISION	COD_SEF	COD_RESTAURANT	COD_JOB
1	15	Petrescu	Marius	email6@yahoo.com	0712345678	9450	12-JAN-21	0.35	11	101	1001
2	13	Neagu	Marian	email4@yahoo.com	0728647009	2625	12-MAR-21	0.1	11	101	1004
3	12	Ionescu	Ana	email3@yahoo.com	0738689839	3150	02-JAN-21	(null)	11	101	1002
4	11	Gheorghe	Mihai	email2@yahoo.com	0728622839	10920	10-DEC-20	0.1	10	101	1005
5	10	Popescu	Ion	email1@yahoo.com	0728647839	25000	02-OCT-20	(null)	(null)	101	1000
6	16	Lazar	Nicoleta	email7@yahoo.com	0711223344	9450	01-FEB-21	0.2	11	101	1001
7	17	Ion	Dan	email8@yahoo.com	0701234985	5250	18-APR-21	(null)	15	101	1006
8	14	Toma	Daniela	email5@yahoo.com	0728678901	2094.75	02-SEP-20	(null)	11	101	1003

- Pentru oras = Busteni și cod manager = 11:

```
DECLARE
v_oras adresa.oras%TYPE := '&oras';
cod_manager angajat.cod_angajat%TYPE := '&cod';
cod_ad adresa.cod_adresa%TYPE;
BEGIN
cod_ad := ex14.codul_adresei(v_oras);
ex14.mareste_salariul(cod_manager, cod_ad);
END;
```

Script Output x | Task completed in 6.698 seconds

Error starting at line : 1,350 in command -

```
DECLARE
v_oras adresa.oras%TYPE := 'Busteni';
cod_manager angajat.cod_angajat%TYPE := '11';
cod_ad adresa.cod_adresa%TYPE;
BEGIN
cod_ad := ex14.codul_adresei(v_oras);
ex14.mareste_salariul(cod_manager, cod_ad);
END;
```

Error report -

ORA-20000: Nu exista manager cu codul transmis ca parametru!

ORA-06512: at "ANDREEA_SORA1.EX14", line 92

ORA-06512: at line 7

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.

- Pentru oras = Mizil și cod manager = 12:

```

DECLARE
v_oras adresa.oras%TYPE := '&oras';
cod_manager angajat.cod_angajat%TYPE := '&cod';
cod_ad adresa.cod_adresa%TYPE;
BEGIN
    cod_ad := ex14.codul_adresei(v_oras);
    ex14.mareste_salariul(cod_manager, cod_ad);
END;

```

Script Output x | Task completed in 14.547 seconds

```

Error starting at line : 1,350 in command -
DECLARE
v_oras adresa.oras%TYPE := 'Mizil';
cod_manager angajat.cod_angajat%TYPE := '15';
cod_ad adresa.cod_adresa%TYPE;
BEGIN
    cod_ad := ex14.codul_adresei(v_oras);
    ex14.mareste_salariul(cod_manager, cod_ad);
END;
Error report -
ORA-20000: Nu exista adresa in orasul transmis ca parametru!
ORA-06512: at "ANDREEA_SORA1.EX14", line 60
ORA-06512: at line 6
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.

```

- Apel pentru funcția nr_ang_cu_2_joburi:

```

BEGIN
dbms_output.put_line('Numarul angajatilor cu 2 job-uri: ' || ex14.nr_ang_cu_2_joburi);
END;

```

```

BEGIN
dbms_output.put_line('Numarul angajatilor cu 2 job-uri: ' || ex14.nr_ang_cu_2_joburi);
END;

```

Script Output x | Task completed in 0.027 seconds

```

PL/SQL procedure successfully completed.

Numarul angajatilor cu 2 job-uri: 3

PL/SQL procedure successfully completed.

```

- Apeluri funcție nr_ang_in_res:

```

BEGIN
dbms_output.put_line('Numarul angajatilor din restaurantul cu codul 101: ' ||
ex14.nr_ang_in_res(101));
END;

```

```
BEGIN
dbms_output.put_line('Numarul angajatilor din restaurantul cu codul 101: ' || ex14.nr_ang_in_res(101));
END;
```

Script Output x
Task completed in 0.045 seconds

PL/SQL procedure successfully completed.

Numarul angajatilor din restaurantul cu codul 101: 8

PL/SQL procedure successfully completed.

```
BEGIN
dbms_output.put_line('Numarul angajatilor din restaurantul cu codul 120: ' ||
ex14.nr_ang_in_res(120));
END;
```

```
BEGIN
dbms_output.put_line('Numarul angajatilor din restaurantul cu codul 120: ' || ex14.nr_ang_in_res(120));
END;
```

Script Output x
Task completed in 0.036 seconds

BEGIN
dbms_output.put_line('Numarul angajatilor din restaurantul cu codul 120: ' || ex14.nr_ang_in_res(120));
END;
Error report -
ORA-20000: Nu exista restaurant cu codul transmis ca parametru!
ORA-06512: at "ANDREEA_SORA1.EX14", line 139
ORA-06512: at line 2
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.

- Apeluri procedura update_salariu:

execute ex14.update_salariu(10, 25700);

```
execute ex14.update_salariu(10, 25700);
```

Script Output x
Task completed in 0.047 seconds

PL/SQL procedure successfully completed.

Update realizat cu succes!

PL/SQL procedure successfully completed.

Înainte de update:

```
select *
from angajat
where cod_angajat = 10;
```

Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0.002 seconds

COD_ANGAJAT	NUME_ANGAJAT	PRENUME_ANGAJAT	EMAIL	NR_TELEFON	SALARIU	DATA_ANGAJARE	COMISION	COD_SEF	COD_RESTAURANT	COD_JOB
1	10 Popescu	Ion	email1@yahoo.com	0728647839	25000	02-OCT-20	(null)	(null)	101	1000

După update:

```
select *
from angajat
where cod_angajat = 10;
```

Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0.001 seconds

COD_ANGAJAT	NUME_ANGAJAT	PRENUME_ANGAJAT	EMAIL	NR_TELEFON	SALARIU	DATA_ANGAJARE	COMISION	COD_SEF	COD_RESTAURANT	COD_JOB
1	10 Popescu	Ion	email1@yahoo.com	0728647839	25700	02-OCT-20	(null)	(null)	101	1000

execute ex14.update_salariu(23, 259);

```
execute ex14.update_salariu(23, 259);
```

Script Output x

Task completed in 0.036 seconds

Salariul nou introdus nu este conform limitelor salariale! Limita inferioara: 10000; Limita superioara: 30000

PL/SQL procedure successfully completed.

- Apeluri procedura concediaza:

execute ex14.concediaza(30);

```
execute ex14.concediaza(30);
```

Script Output x

Task completed in 0.065 seconds

Angajatul 30 a fost concediat!

PL/SQL procedure successfully completed.

Dacă mai rulăm o data codul de mai sus, vom obține:

```
execute ex14.concediaza(30);
```

Script Output x

Task completed in 0.062 seconds

Angajatul 30 a fost deja concediat!

PL/SQL procedure successfully completed.

execute ex14.concediaza(39);

```
execute ex14.concediaza(39);
```

Script Output x

Task completed in 0.064 seconds

Error report -

ORA-20000: Nu exista angajat cu codul transmis ca parametru!

ORA-06512: at "ANDREEA_SORA1.EX14", line 205

ORA-06512: at line 1

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.

BIBLIOGRAFIE

Curs – Baze de Date - Lect. Univ. Dr. Marin Letiția Ana

Curs – Sisteme de gestiune a bazelor de date – Lect. Univ. Dr. Gabriela Mihai