

BĂICOIANU ALEXANDRA-BIANCA  
GRUPA 251

# **MANAGEMENTUL UNEI COMPANII DE LIVRARI**

*–Proiect Sisteme de Gestiune a Bazelor de Date–*

## CUPRINS

- Prezentați pe scurt baza de date (utilitatea ei).
- Realizați diagrama entitate-relație (ERD).
- Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare.
- Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, implementând toate constrângerile de integritate necesare (chei primare, cheile externe etc).
- Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).
- Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze două tipuri de colecție studiate. Apelați subprogramul.
- Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri de cursoare studiate, unul dintre acestea fiind cursor parametrizat. Apelați subprogramul.
- Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.
- Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent 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.
- Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.
- Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.
- Definiți un trigger de tip LDD. Declanșați trigger-ul.
- Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.
- 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).

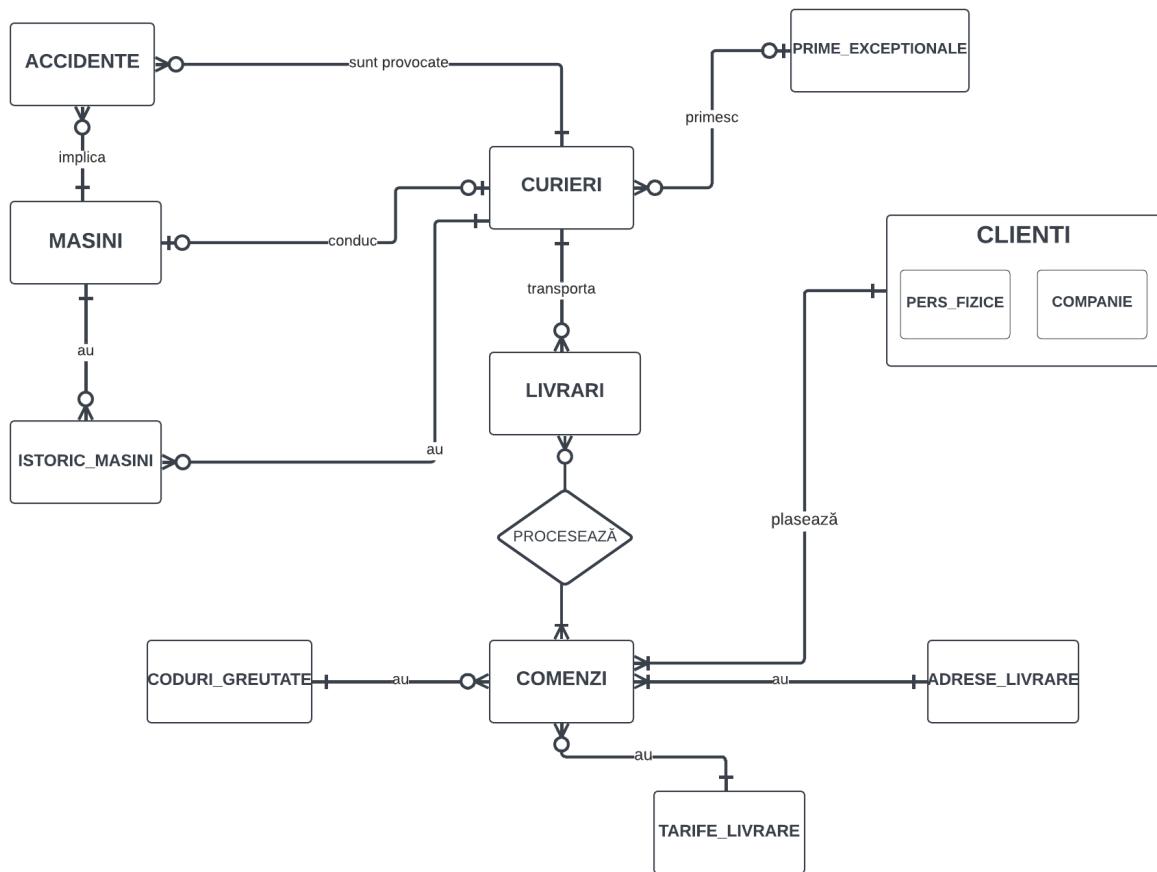
1. Prezentați pe scurt baza de date (utilitatea ei).

În acest proiect am urmărit crearea unei baze de date care să faciliteze gestionarea datelor din cadrul unei companii de curierat, care asigura livrarea comenziilor plasate.

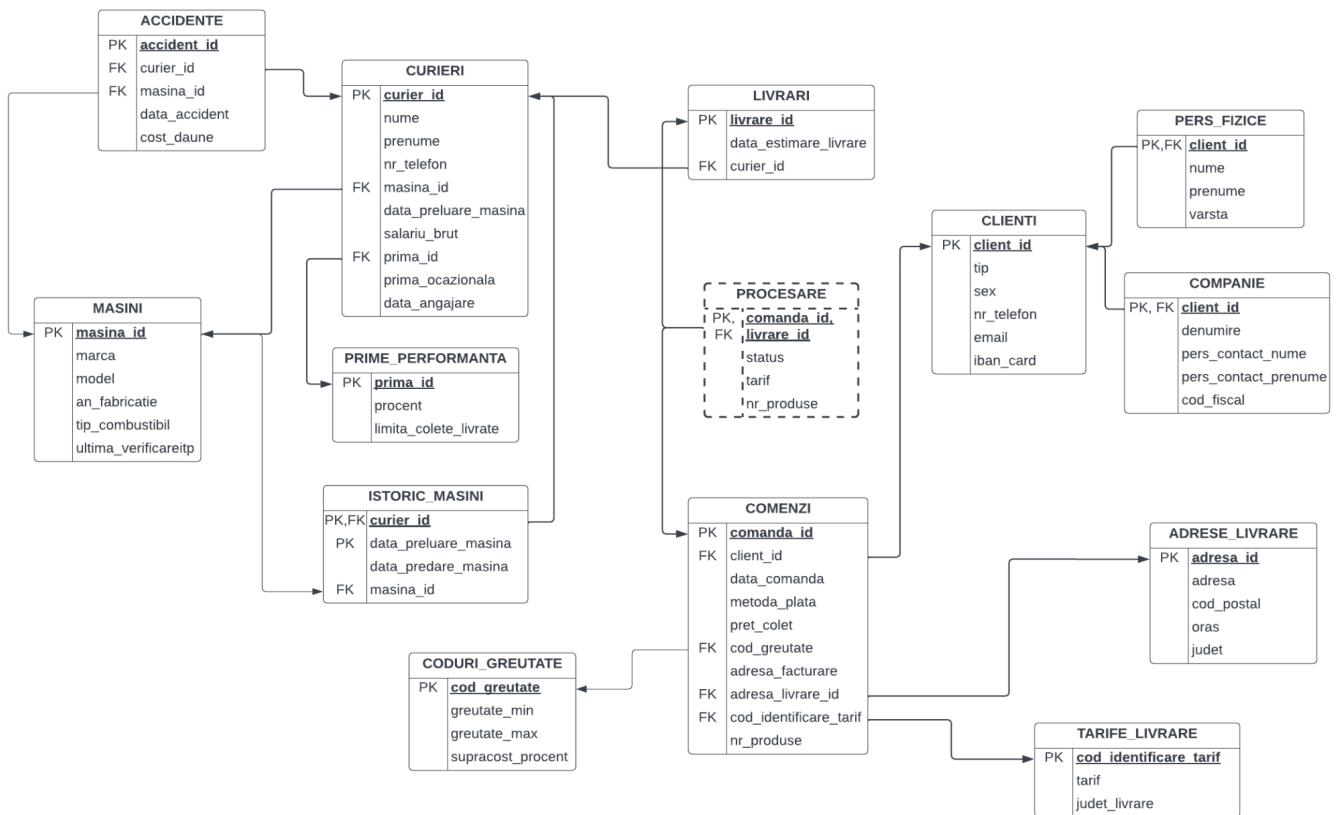
În continuare, voi preciza principiile de funcționare ale acestei companii:

- Compania asigură livrarea produselor către clienți conform comenziilor plasate de aceștia.
- Clientii pot fi persoane fizice sau companii(având o persoana de contact).
- Venitul firmei este dat de costurile de livrare ale comenziilor( diferite în funcție de regiune/județ - “TARIFE\_GREUTATE”), la care se adaugă taxele speciale corespunzătoare greutății coletelor(“CODURI\_GREUTATE”).
- După ce este plasată, o comanda poate fi împărțită în mai multe colete spre a fi livrate
- Coletul este reprezentat prin tabelul “PROCESARE”, legătura dintre tabelele “COMENZI” și “LIVRĂRI”.
- Statusul unui colet poate fi “success”/”failed”; doar coletele cu statusul “success” vor fi livrate de către curieri.
- Un curier poate conduce o singură mașină, dar poate avea un istoric al mașinilor conduse înregistrat în tabelul “ISTORIC\_MASINI”.
- Curierul poate primi o primă de performanță(in afară de alte prime ocazionale) în funcție de numărul de pachete livrate.
- De asemenea, mașinile companiei pot fi implicate în accidente(“ACCIDENTE”). Costurile reparațiilor din urma accidentelor vor fi suportate de către curieri.

2. Realizați diagrama entitate-relație (ERD).



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



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

```
CREATE TABLE TARIFE_LIVRARE
(
cod_identificare_tarif NUMBER NOT NULL PRIMARY KEY,
tarif NUMBER NOT NULL,
judet_livrare VARCHAR2(300) NOT NULL
);

CREATE TABLE PRIME_PERFORMANTA
(
prima_id NUMBER NOT NULL PRIMARY KEY,
procent NUMBER NOT NULL CHECK(procent>0),
limita_colete_livrate NUMBER NOT NULL CHECK(limita_colete_livrate >0)
);

CREATE TABLE CODURI_GREUTATE
(
cod_greutate NUMBER PRIMARY KEY,
greutate_min NUMBER NOT NULL,
greutate_max NUMBER NOT NULL,
supracost_procent NUMBER NOT NULL
);

CREATE TABLE ADRESE_LIVRARE
(
adresa_id NUMBER PRIMARY KEY,
adresa VARCHAR2(200) NOT NULL,
cod_postal VARCHAR2(20) NOT NULL,
oras VARCHAR2(50), -- poate fi dedus din codul postal, deci nu e obligatoriu
judet VARCHAR2(50) NOT NULL
);

CREATE TABLE CLIENTI
(
client_id NUMBER NOT NULL PRIMARY KEY,
tip VARCHAR2(20) NOT NULL CHECK(tip='juridic' OR tip='fizic'),
sex VARCHAR2(10) CHECK(sex IS NULL OR sex='feminin' OR sex='masculin'), --
null in caz de persoana juridica(companie)
nr_telefon VARCHAR2(20) NOT NULL,
email VARCHAR2(50),
iban_card VARCHAR2(150) NOT NULL
);

CREATE TABLE MASINI
(
masina_id NUMBER NOT NULL PRIMARY KEY,
marca VARCHAR2(50) NOT NULL,
model VARCHAR2(50) NOT NULL,
```

```
an_fabricatie NUMBER,
tip_combustibil VARCHAR2(50) CHECK(tip_combustibil IN ('benzina',
'motorina', 'hibrid', 'electric')),
ultima_verificareITP DATE
);

CREATE TABLE CURIERI
(
curier_id NUMBER NOT NULL PRIMARY KEY,
nume VARCHAR2(20) NOT NULL,
prenume VARCHAR2(20) NOT NULL,
nr_telefon VARCHAR2(10) NOT NULL,
masina_id NUMBER, --nu e not null, poate nu i s-a atribuit masina inca
data_preluare_masina DATE,
salariu_brut NUMBER NOT NULL CHECK(salariu_brut>=2580),
prima_id NUMBER, --nu e not null, poate nu are bonus
prima_ocasionala NUMBER, -- nu e not null(EX prima Craciun, prima nunta
etc.)
data_angajare DATE NOT NULL,
CONSTRAINT curieri_masini_fk FOREIGN KEY(masina_id) REFERENCES
MASINI(MASINA_ID) ON DELETE SET NULL,
CONSTRAINT curieri_prima_fk FOREIGN KEY(prima_id) REFERENCES
PRIME_PERFORMANTA(PRIMA_ID) ON DELETE SET NULL
);

CREATE TABLE LIVRARI
(
livrare_id NUMBER PRIMARY KEY,
data_estimare_livrare DATE NOT NULL,
curier_id NUMBER NOT NULL,
CONSTRAINT livrari_courier_fk FOREIGN KEY(curier_id) REFERENCES
curieri(curier_id) ON DELETE CASCADE
);

CREATE TABLE COMENZI
(
comanda_id NUMBER PRIMARY KEY,
client_id NUMBER NOT NULL,
data_comanda DATE NOT NULL,
metoda_plata VARCHAR2(50) NOT NULL CHECK(metoda_plata='card online' OR
metoda_plata='card la primire' OR metoda_plata='cash la primire' OR
metoda_plata='pay pal'),
pret_colet FLOAT NOT NULL, --al pachetului, fara transport si taxe extra
cod_greutate NUMBER NOT NULL,
adresa_facturare VARCHAR2(100) NOT NULL,
adresa_livrare_id NUMBER NOT NULL,
cod_identificare_tarif NUMBER NOT NULL,
CONSTRAINT comenzi_clienti_fk FOREIGN KEY(client_id) REFERENCES
CLIENTI(client_id) ON DELETE CASCADE,
CONSTRAINT comenzi_greutate_fk FOREIGN KEY(cod_greutate) REFERENCES
CODURI_GREUTATE(cod_greutate) ON DELETE CASCADE,
CONSTRAINT comenzi_adresa_livrare_fk FOREIGN KEY(adresa_livrare_id)
REFERENCES ADRESE_LIVRARIE(adresa_id) ON DELETE CASCADE,
```

```
CONSTRAINT comenzi_tarife_livrare_fk FOREIGN KEY(cod_identificare_tarif)
REFERENCES TARIFE_LIVRARE(cod_identificare_tarif) ON DELETE CASCADE
);

CREATE TABLE PERS_FIZICE
(
    client_id NUMBER NOT NULL PRIMARY KEY,
    nume VARCHAR2(50) NOT NULL,
    prenume VARCHAR2(50) NOT NULL,
    varsta NUMBER CHECK(varsta>=16),
    CONSTRAINT clienti_pers_fizice_fk FOREIGN KEY(client_id) REFERENCES
CLIENTI(client_id) ON DELETE CASCADE
);

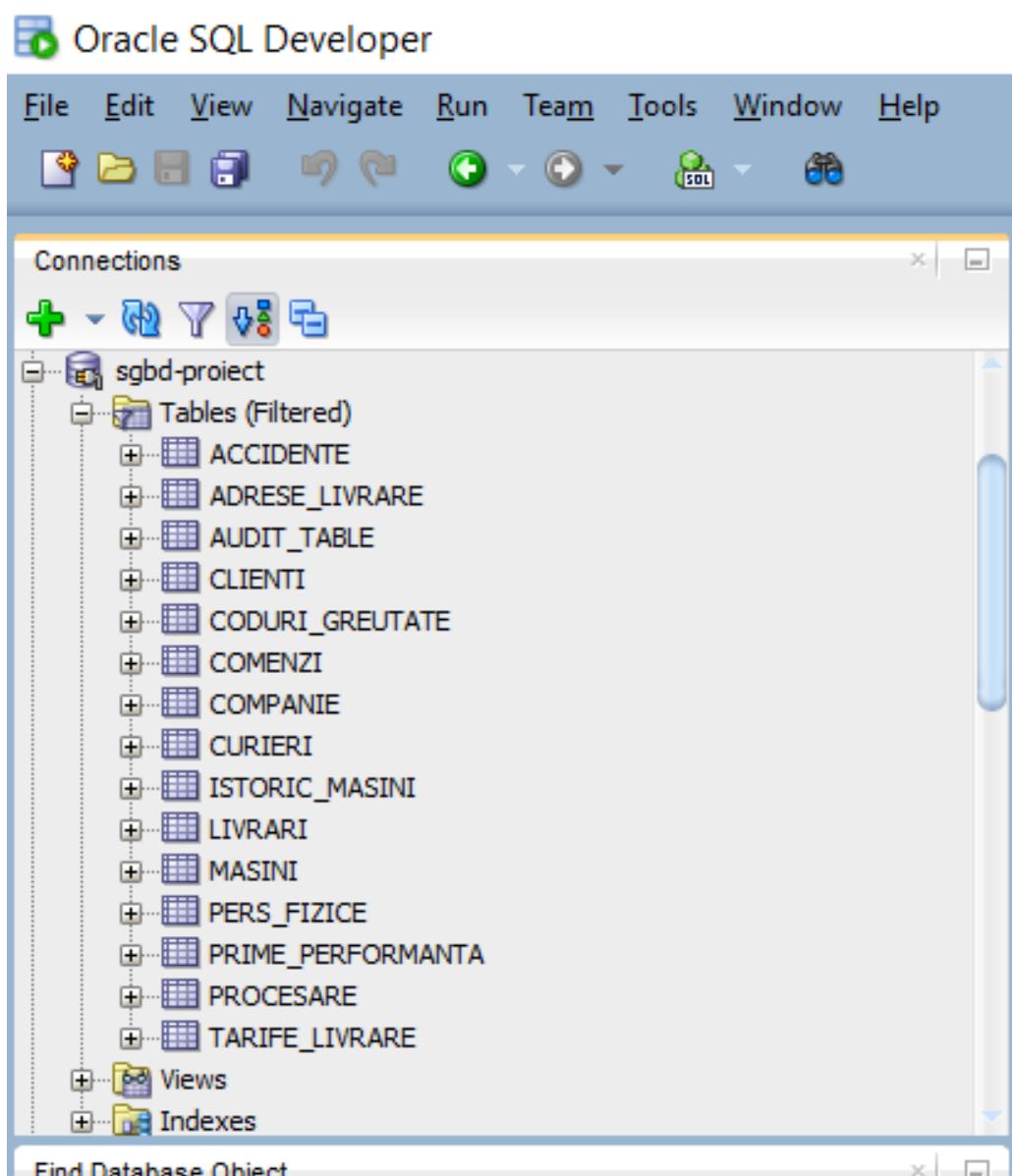
CREATE TABLE COMANIE
(
    client_id NUMBER NOT NULL PRIMARY KEY,
    denumire VARCHAR2(100) NOT NULL,
    pers_contact_nume VARCHAR2(20) NOT NULL,
    pers_contact_prenume VARCHAR2(20) NOT NULL,
    cod_fiscal VARCHAR2(50) NOT NULL,
    CONSTRAINT clienti_companie_fk FOREIGN KEY(client_id) REFERENCES
CLIENTI(client_id) ON DELETE CASCADE
);

CREATE TABLE ISTORIC_MASINI
(
    curier_id NUMBER,
    data_preluare_masina DATE NOT NULL,
    data_predare_masina DATE NOT NULL,
    masina_id NUMBER NOT NULL,
    PRIMARY KEY(curier_id, data_preluare_masina),
    CONSTRAINT istoric_masini_masini_fk FOREIGN KEY(masina_id) REFERENCES
MASINI(masina_id) ON DELETE CASCADE,
    CONSTRAINT istoric_masini_curieri_fk FOREIGN KEY(curier_id) REFERENCES
CURIERI(curier_id) ON DELETE CASCADE
);

CREATE TABLE ACCIDENTE
(
    accident_id NUMBER NOT NULL PRIMARY KEY,
    curier_id NUMBER NOT NULL,
    masina_id NUMBER NOT NULL,
    data_accident DATE NOT NULL,
    cost_daune NUMBER NOT NULL,
    CONSTRAINT accidente_masini_fk FOREIGN KEY(masina_id) REFERENCES
MASINI(masina_id) ON DELETE CASCADE,
    CONSTRAINT accidente_curieri_fk FOREIGN KEY(curier_id) REFERENCES
CURIERI(curier_id) ON DELETE CASCADE
);

CREATE TABLE PROCESARE
(
```

```
comanda_id NUMBER NOT NULL,  
livrare_id NUMBER NOT NULL,  
status VARCHAR2(10) CHECK(status IS NULL OR status='success' OR  
status='failed'),  
tarif FLOAT NOT NULL,  
nr_produse NUMBER,  
PRIMARY KEY(livrare_id, comanda_id),  
CONSTRAINT procesare_livrari_fk FOREIGN KEY(livrare_id) REFERENCES  
LIVRARI(livrare_id) ON DELETE CASCADE,  
CONSTRAINT procesare_comenzi_fk FOREIGN KEY(comanda_id) REFERENCES  
COMENZI(comanda_id) ON DELETE CASCADE  
);
```



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

```
INSERT INTO TARIFE_LIVRARE VALUES (1, 7, 'Prahova');
INSERT INTO TARIFE_LIVRARE VALUES (2, 12, 'Brașov');
INSERT INTO TARIFE_LIVRARE VALUES (3, 16, 'Constanța');
INSERT INTO TARIFE_LIVRARE VALUES (4, 18, 'Craiova');
INSERT INTO TARIFE_LIVRARE VALUES (5, 25, 'Botoșani');
INSERT INTO TARIFE_LIVRARE VALUES (6, 30, 'Maramureș');
INSERT INTO TARIFE_LIVRARE VALUES (7, 27, 'Cluj');
INSERT INTO TARIFE_LIVRARE VALUES (8, 15, 'Harghita');
INSERT INTO TARIFE_LIVRARE VALUES (9, 5, 'Ilfov');
INSERT INTO TARIFE_LIVRARE VALUES (10, 10, 'Brăila');
```

COD_IDENTIFICARE_TARIF	TARIF	JUDET_LIVRARE
1	1	7 Prahova
2	2	12 Brașov
3	3	16 Constanța
4	4	18 Craiova
5	5	25 Botoșani
6	6	30 Maramureș
7	7	27 Cluj
8	8	15 Harghita
9	9	5 Ilfov
10	10	Brăila

```
INSERT INTO PRIME_PERFORMANTA VALUES (1, 2, 50);
INSERT INTO PRIME_PERFORMANTA VALUES (2, 5, 100);
INSERT INTO PRIME_PERFORMANTA VALUES (3, 10, 200);
INSERT INTO PRIME_PERFORMANTA VALUES (4, 15, 300);
INSERT INTO PRIME_PERFORMANTA VALUES (5, 25, 500);
```

PRIMA_ID	PROCENT	LIMITA_COLETE_LIVRARE
1	1	2 50
2	2	5 100
3	3	10 200
4	4	15 300
5	5	25 500

```
INSERT INTO CODURI_GREUTATE VALUES (1, 0, 2, 0);
INSERT INTO CODURI_GREUTATE VALUES (2, 3, 5, 3);
INSERT INTO CODURI_GREUTATE VALUES (3, 6, 10, 5);
INSERT INTO CODURI_GREUTATE VALUES (4, 11, 20, 10);
INSERT INTO CODURI_GREUTATE VALUES (5, 21, 50, 20);
```

COD_GREUTATE	GREUTATE_MIN	GREUTATE_MAX	SUPRACOST_PROCENT
1	1	0	2
2	2	3	5
3	3	6	10
4	4	11	20
5	5	21	50

```

INSERT INTO ADRESE_LIVRAR VALUES (1001, 'Str Florilor, nr 3', '353436',
'Buftea', 'Ilfov');
INSERT INTO ADRESE_LIVRAR VALUES (1002, 'Aleea Regelui, nr 101', '132452',
'Sinaia', 'Prahova');
INSERT INTO ADRESE_LIVRAR VALUES (1003, 'Str Schelelor, nr 3', '463632',
'Craiova', 'Dolj');
INSERT INTO ADRESE_LIVRAR VALUES (1004, 'Str Castanilor, bl A6, ap 6',
'963425', 'Botoșani', 'Botoșani');
INSERT INTO ADRESE_LIVRAR VALUES (1005, 'Strada Eruptiei, nr 10', '142400',
'Cluj-Napoca', 'Cluj');
INSERT INTO ADRESE_LIVRAR VALUES (1006, 'Aleea Rozelor, nr 19', '105600',
'Câmpina', 'Prahova');
INSERT INTO ADRESE_LIVRAR VALUES (1007, 'Cartier Nicolae Grigorescu, Str 1
Decembrie, nr 7', '310553', 'Miercurea-Ciuc', 'Harghita');
INSERT INTO ADRESE_LIVRAR VALUES (1008, 'Bulevardul Culturii, nr 101',
'580132', 'Mangalia', 'Constanța');
INSERT INTO ADRESE_LIVRAR VALUES (1009, 'Str Ecaterina Teodoroiu, bl B6, ap
14', '142024', 'Brăila', 'Brăila');
INSERT INTO ADRESE_LIVRAR VALUES (1010, 'Bulevardul Carol I, nr 21',
'100892', 'Constanța', 'Constanța');

```

ADRESA_ID	ADRESA	COD_POSTAL	ORAS	JUDET
1	1001 Str Florilor, nr 3	353436	Buftea	Ilfov
2	1002 Aleea Regelui, nr 101	132452	Sinaia	Prahova
3	1003 Str Schelelor, nr 3	463632	Craiova	Dolj
4	1004 Str Castanilor, bl A6, ap 6	963425	Botosani	Botosani
5	1005 Strada Eruptiei, nr 10	142400	Cluj-Napoca	Cluj
6	1006 Aleea Rozelor, nr 19	105600	Câmpina	Prahova
7	1007 Cartier Nicolae Grigorescu, Str 1 Decembrie, nr 7	310553	Miercurea-Ciuc	Harghita
8	1008 Bulevardul Culturii, nr 101	580132	Mangalia	Constanța
9	1009 Str Ecaterina Teodoroiu, bl B6, ap 14	142024	Brăila	Brăila
10	1010 Bulevardul Carol I, nr 21	100892	Constanta	Constanta

```

INSERT INTO MASINI VALUES (21, 'Audi', 'A3', 2018, 'benzina',
TO_DATE('2022-01-06', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO MASINI VALUES (22, 'Dacia', 'Logan', 2005, 'motorina',
TO_DATE('2022-03-07', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO MASINI VALUES (23, 'Renault', 'Clio', 2011, 'motorina',
TO_DATE('2021-12-29', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO MASINI VALUES (24, 'Volkswagen', 'Golf 7', 2020, 'hibrid',
TO_DATE('2022-06-25', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO MASINI VALUES (25, 'Volkswagen', 'Polo', 2009, 'benzina',
TO_DATE('2022-08-17', 'YYYY-MM-DD HH24:MI:SS'));

```

MASINA_ID	MARCA	MODEL	AN_FABRICATIE	TIP_COMBUSTIBIL	ULTIMA_VERIFICARE_TIP
1	21 Audi	A3	2018	benzina	06-01-2022
2	22 Dacia	Logan	2005	motorina	07-03-2022
3	23 Renault	Clio	2011	motorina	29-12-2021
4	24 Volkswagen	Golf 7	2020	hibrid	25-06-2022
5	25 Volkswagen	Polo	2009	benzina	17-08-2022

```
INSERT INTO CURIERI VALUES (31, 'Popescu', 'Marian', '0724562517', 23,
TO_DATE('2022-06-12', 'YYYY-MM-DD HH24:MI:SS'), 3200, 1, 100,
TO_DATE('2016-06-07', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO CURIERI VALUES (32, 'Ionescu', 'Marius', '0726257252', 21,
TO_DATE('2021-11-28', 'YYYY-MM-DD HH24:MI:SS'), 2950, 3, 270,
TO_DATE('2014-06-19', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO CURIERI VALUES (33, 'Mocanu', 'Dorel', '0742527526', 25,
TO_DATE('2022-10-25', 'YYYY-MM-DD HH24:MI:SS'), 3600, null, null,
TO_DATE('2020-06-22', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO CURIERI VALUES (34, 'Andronic', 'Vasile', '0752426523', 24,
TO_DATE('2022-10-06', 'YYYY-MM-DD HH24:MI:SS'), 3300, 4, 150,
TO_DATE('2019-10-29', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO CURIERI VALUES (35, 'Ilie ', 'Alexandru', '0724265272', 22,
TO_DATE('2022-11-19', 'YYYY-MM-DD HH24:MI:SS'), 3100, 5, null,
TO_DATE('2019-11-17', 'YYYY-MM-DD HH24:MI:SS'));
```

CURIER_ID	NUME	PRENUME	NR_TELEFON	MASINA_ID	DATA_PRELUCARE_MASINA	SALARIU_BRUT	PRIMA_ID	PRIMA_OCAZIONALA	DATA_ANGAJARE
1	31 Popescu	Marian	0724562517	23	12-06-2022	3200	1	100	07-06-2016
2	32 Ionescu	Marius	0726257252	21	28-11-2021	2950	3	270	19-06-2014
3	33 Mocanu	Dorel	0742527526	25	25-10-2022	3600	(null)	22-06-2020	
4	34 Andronic	Vasile	0752426523	24	06-10-2022	3300	4	150	29-10-2019
5	35 Ilie	Alexandru	0724265272	22	19-11-2022	3100	5	(null)	17-11-2019

```
INSERT INTO ISTORIC_MASINI VALUES (31, TO_DATE('2022-01-05', 'YYYY-MM-DD
HH24:MI:SS'), TO_DATE('2022-05-07', 'YYYY-MM-DD HH24:MI:SS'), 21);
INSERT INTO ISTORIC_MASINI VALUES (31, TO_DATE('2020-06-08', 'YYYY-MM-DD
HH24:MI:SS'), TO_DATE('2021-12-29', 'YYYY-MM-DD HH24:MI:SS'), 22);
INSERT INTO ISTORIC_MASINI VALUES (32, TO_DATE('2017-05-08', 'YYYY-MM-DD
HH24:MI:SS'), TO_DATE('2019-06-23', 'YYYY-MM-DD HH24:MI:SS'), 23);
INSERT INTO ISTORIC_MASINI VALUES (33, TO_DATE('2020-09-25', 'YYYY-MM-DD
HH24:MI:SS'), TO_DATE('2021-11-26', 'YYYY-MM-DD HH24:MI:SS'), 21);
INSERT INTO ISTORIC_MASINI VALUES (35, TO_DATE('2017-10-22', 'YYYY-MM-DD
HH24:MI:SS'), TO_DATE('2022-12-01', 'YYYY-MM-DD HH24:MI:SS'), 25);
```

CURIER_ID	DATA_PRELUCARE_MASINA	DATA_PREDARE_MASINA	MASINA_ID
1	31 05-01-2022	07-05-2022	21
2	31 08-06-2020	29-12-2021	22
3	32 08-05-2017	23-06-2019	23
4	33 25-09-2020	26-11-2021	21
5	35 22-10-2017	01-12-2022	25

```
INSERT INTO ACCIDENTE VALUES (1, 31, 23, TO_DATE('2021-10-07', 'YYYY-MM-DD
HH24:MI:SS'), 700);
INSERT INTO ACCIDENTE VALUES (2, 32, 21, TO_DATE('2022-02-13', 'YYYY-MM-DD
HH24:MI:SS'), 2800);
INSERT INTO ACCIDENTE VALUES (3, 33, 25, TO_DATE('2022-07-06', 'YYYY-MM-DD
HH24:MI:SS'), 7500);
```

```
INSERT INTO ACCIDENTE VALUES (4, 34, 24, TO_DATE('2020-11-08', 'YYYY-MM-DD
HH24:MI:SS'), 500);
INSERT INTO ACCIDENTE VALUES (5, 35, 22, TO_DATE('2021-06-07', 'YYYY-MM-DD
HH24:MI:SS'), 1000);
```

ACCIDENT_ID	CURIER_ID	MASINA_ID	DATA_ACCIDENT	COST_DAUNE
1	1	31	23 07-10-2021	700
2	2	32	21 13-02-2022	2800
3	3	33	25 06-07-2022	7500
4	4	34	24 08-11-2020	500
5	5	35	22 07-06-2021	1000

```
INSERT INTO CLIENTI VALUES (10001, 'fizic', 'feminin', '0742671524',
'ioana.popescu@gmail.com', 'ING 7982 1222 4850 2840');
INSERT INTO CLIENTI VALUES (10002, 'juridic', null, '0756267214',
'dobal.design@gmail.com', 'BCR 1111 1111 1111 2121');
INSERT INTO CLIENTI VALUES (10003, 'fizic', 'masculin', '025325263',
'andrei_mirel@yahoo.com', 'BRD 8509 9999 6748 2829');
INSERT INTO CLIENTI VALUES (10004, 'juridic', null, '074521423',
'aaannnaaa@yahoo.com', 'ING 7982 1222 5372 2840');
INSERT INTO CLIENTI VALUES (10005, 'juridic', null, '072315364',
'contact@daspi.ro', 'BCR 1313 1212 1414 1515');
INSERT INTO CLIENTI VALUES (10006, 'fizic', 'feminin', '072424253',
'iordache_betty@yahoo.com', 'BRD 7474 2325 6748 2829');
INSERT INTO CLIENTI VALUES (10007, 'juridic', null, '024354646',
'contact@cameron.ro', 'ING 9263 6836 4850 2840');
INSERT INTO CLIENTI VALUES (10008, 'juridic', null, '073425232',
'contact@autonom.ro', 'BCR 2222 2222 2222 2121');
INSERT INTO CLIENTI VALUES (10009, 'fizic', 'feminin', '074132536',
'dobrescu_miruna@gmail.com', 'BRD 5378 2537 2638 1627');
INSERT INTO CLIENTI VALUES (10010, 'fizic', 'masculin', '072314242',
'ionescu_adrian@gmail.com', 'ING 7982 1222 1638 1182');
```

CLIENT_ID	TIP	SEX	NR_TELEFON	EMAIL	IBAN_CARD
1	10001	fizic	feminin	0742671524	ioana.popescu@gmail.com
2	10002	juridic	(null)	0756267214	dobal.design@gmail.com
3	10003	fizic	masculin	025325263	andrei_mirel@yahoo.com
4	10004	juridic	(null)	074521423	aaannnaaa@yahoo.com
5	10005	juridic	(null)	072315364	contact@daspi.ro
6	10006	fizic	feminin	072424253	iordache_betty@yahoo.com
7	10007	juridic	(null)	024354646	contact@cameron.ro
8	10008	juridic	(null)	073425232	contact@autonom.ro
9	10009	fizic	feminin	074132536	dobrescu_miruna@gmail.com
10	10010	fizic	masculin	072314242	ionescu_adrian@gmail.com

```
INSERT INTO PERS_FIZICE VALUES (10001, 'Popescu', 'Ioana', 21);
INSERT INTO PERS_FIZICE VALUES (10003, 'Ionescu', 'Mirel', 53);
INSERT INTO PERS_FIZICE VALUES (10006, 'Iordache', 'Beatrice', null);
INSERT INTO PERS_FIZICE VALUES (10009, 'Popescu', 'Miruna', 34);
INSERT INTO PERS_FIZICE VALUES (10010, 'Ionescu', 'Adrian', 27);
```

CLIENT_ID	NUME	PRENUME	VARSTA
1	10001	Popescu	Ioana
2	10003	Ionescu	Mirel
3	10006	Iordache	Beatrice (null)
4	10009	Popescu	Miruna
5	10010	Ionescu	Adrian

```

INSERT INTO COMANIE VALUES (10002, 'Dobal Design', 'Dobre', 'Valentin',
'131314');
INSERT INTO COMANIE VALUES (10004, 'Goodies Catering', 'Teodorescu ', 'Ana',
'121262');
INSERT INTO COMANIE VALUES (10005, 'SC DASPI', 'Spânu', 'Andrada',
'152563');
INSERT INTO COMANIE VALUES (10007, 'CAMERON SRL', 'Rus', 'Alina', '214252');
INSERT INTO COMANIE VALUES (10008, 'Autonom ', 'Marin', 'Iustin', '213233');

```

CLIENT_ID	DENUMIRE	PERS_CONTACT_NUME	PERS_CONTACT_PRENUME	COD_FISCAL
1	10002 Dobal Design	Dobre	Valentin	131314
2	10004 Goodies Catering	Teodorescu	Ana	121262
3	10005 SC DASPI	Spânu	Andrada	152563
4	10007 CAMERON SRL	Rus	Alina	214252
5	10008 Autonom	Marin	Iustin	213233

```

INSERT INTO LIVRARI VALUES (70, TO_DATE('2022-11-29', 'YYYY-MM-DD
HH24:MI:SS'), 31);
INSERT INTO LIVRARI VALUES (71, TO_DATE('2022-12-06', 'YYYY-MM-DD
HH24:MI:SS'), 34);
INSERT INTO LIVRARI VALUES (72, TO_DATE('2022-12-29', 'YYYY-MM-DD
HH24:MI:SS'), 35);
INSERT INTO LIVRARI VALUES (73, TO_DATE('2022-12-20', 'YYYY-MM-DD
HH24:MI:SS'), 32);
INSERT INTO LIVRARI VALUES (74, TO_DATE('2022-12-23', 'YYYY-MM-DD
HH24:MI:SS'), 34);

```

LIVRARE_ID	DATA_ESTIMARE_LIVRARE	CURIER_ID
1	70 29-11-2022	31
2	71 06-12-2022	34
3	72 29-12-2022	35
4	73 20-12-2022	32
5	74 23-12-2022	34

```

INSERT INTO COMENZI VALUES (1, 10010, TO_DATE('2022-11-23', 'YYYY-MM-DD
HH24:MI:SS'), 'card online', 350, 2, 'Str Florilor, nr 3', 1001, 9, 1);
INSERT INTO COMENZI VALUES (2, 10002, TO_DATE('2022-10-18', 'YYYY-MM-DD
HH24:MI:SS'), 'card la primire', 980, 1, 'Str Castanilor, bl A6, ap 6', 1004,
5, 2);
INSERT INTO COMENZI VALUES (3, 10001, TO_DATE('2022-11-30', 'YYYY-MM-DD
HH24:MI:SS'), 'card online', 1000, 5, 'Cartier Nicolae Grigorescu, Str 1
Decembrie, nr 7', 1007, 8, 3);

```

```
INSERT INTO COMENZI VALUES (4, 10008, TO_DATE('2022-12-08', 'YYYY-MM-DD
HH24:MI:SS'), 'card online', 720, 5, 'Str Florilor, nr 3', 1001, 9, 3);
INSERT INTO COMENZI VALUES (5, 10007, TO_DATE('2022-12-31', 'YYYY-MM-DD
HH24:MI:SS'), 'card la primire', 4260, 4, 'Strada Eruptiei, nr 10', 1005, 7,
2);
INSERT INTO COMENZI VALUES (6, 10006, TO_DATE('2022-12-02', 'YYYY-MM-DD
HH24:MI:SS'), 'card online', 1230, 1, 'Bulevardul Carol I, nr 21', 1010, 3,
4);
INSERT INTO COMENZI VALUES (7, 10008, TO_DATE('2022-11-30', 'YYYY-MM-DD
HH24:MI:SS'), 'pay pal', 420, 3, 'Aleea Regelui, nr 101', 1002, 1, 5);
INSERT INTO COMENZI VALUES (8, 10002, TO_DATE('2022-12-05', 'YYYY-MM-DD
HH24:MI:SS'), 'card online', 120, 2, 'Aleea Rozelor, nr 19', 1006, 1, 3);
INSERT INTO COMENZI VALUES (9, 10004, TO_DATE('2022-12-13', 'YYYY-MM-DD
HH24:MI:SS'), 'card online', 280, 1, 'Strada Eruptiei, nr 10', 1005, 7, 1);
INSERT INTO COMENZI VALUES (10, 10009, TO_DATE('2022-12-06', 'YYYY-MM-DD
HH24:MI:SS'), 'cash la primire', 320, 3, 'Bulevardul Culturii, nr 101', 1008,
3, 2);
```

	COMANDA_ID	CLIENT_ID	DATA_COMANDA	METODA_PLATA	PRET_COLET	COD_GRESITATE	ADRESA_FACTURARE	ADRESA_LIVRARE_ID	COD_IDENITIFICARE_TARIF	NR_PRODUSE
1	1	10010	23-11-2022	card online	350	2	Str Florilor, nr 3	1001	9	1
2	2	10002	18-10-2022	card la primire	980	1	Str Castanilor, bl A6, ap 6	1004	5	2
3	3	10001	30-11-2022	card online	1000	5	Cartier Nicolae Grigorescu, Str 1 Decembrie, nr 7	1007	8	3
4	4	10008	08-12-2022	card online	720	5	Str Florilor, nr 3	1001	9	3
5	5	10007	31-12-2022	card la primire	4260	4	Strada Eruptiei, nr 10	1005	7	2
6	6	10006	02-12-2022	card online	1230	1	Bulevardul Carol I, nr 21	1010	3	4
7	7	10008	30-11-2022	pav pal	420	3	Aleea Regelui, nr 101	1002	1	5
8	8	10002	05-12-2022	card online	120	2	Aleea Rozelor, nr 19	1006	1	3
9	9	10004	13-12-2022	card online	280	1	Strada Eruptiei, nr 10	1005	7	1
10	10	10009	06-12-2022	cash la primire	320	3	Bulevardul Culturii, nr 101	1008	3	2

```
INSERT INTO PROCESARE VALUES (1, 70, 'failed', 350, 2);
INSERT INTO PROCESARE VALUES (1, 71, 'success', 350, 1);
INSERT INTO PROCESARE VALUES (2, 71, 'failed', 980, 2);
INSERT INTO PROCESARE VALUES (3, 74, null, 1000, 3);
INSERT INTO PROCESARE VALUES (4, 73, 'success', 720, 1);
INSERT INTO PROCESARE VALUES (5, 72, 'success', 4260, 5);
INSERT INTO PROCESARE VALUES (6, 71, 'success', 1230, 2);
INSERT INTO PROCESARE VALUES (7, 72, null, 420, 1);
INSERT INTO PROCESARE VALUES (8, 74, 'failed', 120, 1);
INSERT INTO PROCESARE VALUES (9, 73, 'success', 280, 2);
INSERT INTO PROCESARE VALUES (10, 70, null, 320, 4);
```

	COMANDA_ID	LIVRARE_ID	STATUS	TARIF	NR_PRODUSE
1	1	70	failed	350	2
2	1	71	success	350	1
3	2	71	failed	980	2
4	3	74	(null)	1000	3
5	4	73	success	720	1
6	5	72	success	4260	5
7	6	71	success	1230	2
8	7	72	(null)	420	1
9	8	74	failed	120	1
10	9	73	success	280	2
11	10	70	(null)	320	4

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

- **Pentru fiecare județ în care s-a plasat o comandă, afișați numărul de comenzi plasate, numărul de orașe în care au fost plasate acele comenzi(pot fi mai multe comenzi plasate în același oraș) și numele orașelor.**

```

CREATE OR REPLACE PROCEDURE ex_colectii IS
    TYPE t_judete IS TABLE OF adrese_livrare.judet%TYPE INDEX BY PLS_INTEGER;
    --index-by-table
    v_judete t_judete;
    TYPE t_nr_comenzi IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
    --index-by-table
    v_nr_comenzi t_nr_comenzi;
    TYPE t_orase IS TABLE OF ADRESE_LIVRARE.oras%TYPE; --nested table
    TYPE t_total_orase IS TABLE OF t_orase; --nested table
    v_orase t_total_orase:=t_total_orase();
    nr_judete NUMBER;
    nr_orase NUMBER;
BEGIN
    SELECT DISTINCT ad.judet, COUNT(*)
    BULK COLLECT INTO v_judete, v_nr_comenzi
    FROM comenzi c JOIN ADRESE_LIVRARE ad ON c.adresa_livrare_id=ad.ADRESA_id
    GROUP BY ad.judet;

    nr_judete:=v_judete.COUNT;

    FOR i IN 1..nr_judete LOOP -- iteram prin judete
        v_orase.EXTEND;

        SELECT DISTINCT ad.oras
        BULK COLLECT INTO v_orase(i)
        FROM comenzi c JOIN ADRESE_LIVRARE ad ON
        c.adresa_livrare_id=ad.adresa_id
        WHERE ad.judet=v_judete(i);
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Detalii comenzi pe județe: ');

    -- pentru fiecare județ, afișam nr de comenzi și nr de orașe(+numele)
    FOR i IN 1..nr_judete LOOP
        DBMS_OUTPUT.PUT(v_judete(i) || ' are ' || v_nr_comenzi(i));
        IF v_nr_comenzi(i)=1 THEN
            DBMS_OUTPUT.PUT(' comanda în ');
        ELSE
            DBMS_OUTPUT.PUT(' comenzi în ');
        END IF;

        DBMS_OUTPUT.PUT(v_orase(i).COUNT);
        IF v_orase(i).COUNT=1 THEN

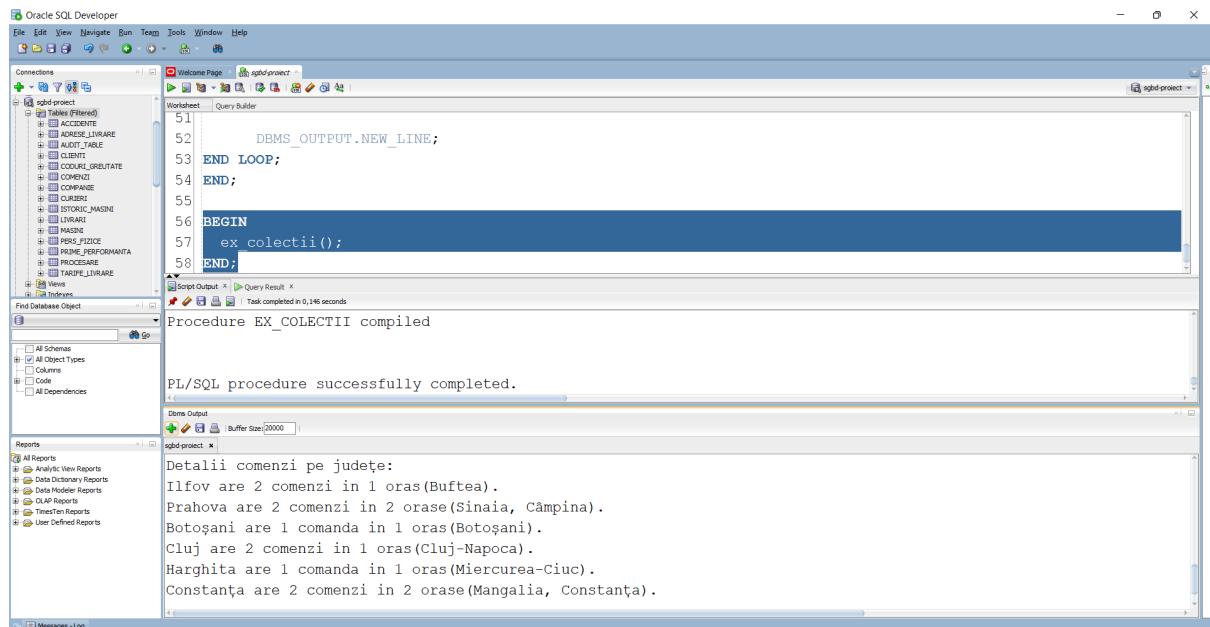
```

```
DBMS_OUTPUT.PUT(' oras(');
ELSE
    DBMS_OUTPUT.PUT(' orase(');
END IF;

nr_orase:=v_orase(i).COUNT;
FOR j IN 1..nr_orase-1 LOOP
    DBMS_OUTPUT.PUT( v_orase(i)(j) || ', ');
END LOOP;
DBMS_OUTPUT.PUT( v_orase(i)(nr_orase) || ') .');

DBMS_OUTPUT.NEW_LINE;
END LOOP;
END;

BEGIN
    ex_colectii();
END;
```



7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri de cursoare studiate, unul dintre acestea fiind cursor parametrizat. Apelați subprogramul.

- **Compania dorește sa realizeze o statistica cu top 3 cele mai grave accidente(conform costurilor daunelor) - data accidentului, șoferul responsabil și daunele materiale**
- **Ulterior, afișați mașinile curierilor care au fost implicate în accidente intr-un anumit an introdus de la tastatura(id, marcă, model), împreuna cu luna accidentului și date despre șofer( id, nume, prenume curier), specificand dacă este masina curentă sau nu.**

```

CREATE OR REPLACE PROCEDURE ex_cursoare IS

an number(4) := &p_an;
-- cursor parametrizat
CURSOR cl_ex_cursoare(an NUMBER) IS
    SELECT c.curier_id curier_id, c.prenume prenume, c.nume nume,
a.data_accident data_accident, a.cost_daune cost_daune, m.masina_id
masina_id,
        m.marca marca, m.model model, c.data_angajare data_angajare,
istm.data_preluare_masina data_preluare_masina, istm.data_predare_masina,
c.data_preluare_masina data_preluare_masina2
    FROM curieri c JOIN accidente a ON c.curier_id=a.curier_id
                JOIN masini m ON a.masina_id=m.masina_id
                LEFT JOIN istoric_masini istm ON
(a.masina_id=istm.masina_id AND a.curier_id=istm.curier_id
                AND
a.data_accident<=istm.DATA_PREDARE_MASINA AND
a.DATA_ACCIDENT>=istm.DATA_PRELUCARE_MASINA )
    WHERE EXTRACT(YEAR FROM a.data_accident) = an;

curenta VARCHAR2(50);
top number(1) := 0;

BEGIN
DBMS_OUTPUT.PUT_LINE('Top 3 accidente 2022 conform pagubelor');
--ciclul cursor cu subcereri
FOR i IN (SELECT c.curier_id id, c.prenume p, c.nume n, a.data_accident da,
a.cost_daune cd, c.salariu_brut sb
    FROM curieri c JOIN accidente a ON c.curier_id=a.curier_id
    ORDER BY a.cost_daune DESC)
LOOP
    DBMS_OUTPUT.PUT_LINE('Data accident: ' || i.da || '; Sofer
responsabil:' || i.id || '; Daune materiale:' || i.cd);
    top := top+1;
    EXIT WHEN top = 3;
END LOOP;

```

```
DBMS_OUTPUT.PUT_LINE('Accidente inregistrate in anul ' || an);
FOR i IN c1_ex_cursoare(an) LOOP --ciclu cursor

    IF i.data_predare_masina < i.data_accident THEN
        curenta:='masina curenta a curierului ';
    ELSE
        curenta:='masina anterioara a curierului ';
    END IF;
    DBMS_OUTPUT.PUT_LINE('Masina cu numarul de identificare(id) ' ||
i.masina_id || ' (' || i.marca || ' ' || i.model || ',' || curenta || i.nume ||
' ' || i.prenume || ', avand id-ul ' || i.curier_id || ', a fost implicata
intr-un accident in luna ' || extract(month from i.data_accident) || '.');
END LOOP;

END;
/


BEGIN
    ex_cursoare();
END;
```

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

- Worksheet:** Displays the PL/SQL code for the procedure `EX_CURSOARE`, specifically the block that calls `ex_cursoare()`.
- Query Result:** Shows the message "Procedure EX\_CURSOARE compiled".
- PL/SQL procedure successfully completed.**
- Logs Output:** Displays the output of the procedure execution, including:
  - Top 3 accidents causing damages:
    - Data accident: 06-07-2022; Sofer responsabil:33; Daune materiale:7500
    - Data accident: 13-02-2022; Sofer responsabil:32; Daune materiale:2800
    - Data accident: 07-06-2021; Sofer responsabil:35; Daune materiale:1000
  - Accidents registered in the year 2021.
  - Mention of accidents involving specific cars (Dacia Logan and Renault Clio).
- Messages Log:** Shows the log messages for the procedure execution.

8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

- **Determinati numărul de comenzi întregi livrate de un anume curier al cărui id este dat. O comanda se consideră întreaga dacă, indiferent dacă a fost procesată în mai multe colete spre livrare, toate au primit statusul 'success'.**
- **Pentru cazul în care curierul respectiv nu are deloc comenzi întregi repartizate pentru a fi livrate, sa se afișeze un mesaj corespunzător.**

```
CREATE OR REPLACE FUNCTION ex_functie(curier curieri.curier_id%TYPE) RETURN NUMBER IS

CURSOR livrari_total(c_id curieri.curier_id%TYPE) IS
-- cursor 1 pentru livrarile efectuate de un curier c_id
    SELECT livrare_id
    FROM livrari
    WHERE curier_id=c_id;

CURSOR comenzi_procesate_succes(c_id curieri.curier_id%TYPE) IS
-- cursor 2 pentru comenziile procesate 'success'
    SELECT com.comanda_id comanda_id, SUM(p.NR_PRODUSE) nr_produse
    FROM comenzi com JOIN procesare p ON com.comanda_id=p.comanda_id
        JOIN livrari l ON l.LIVRAR_ID=p.LIVRAR_ID
    WHERE p.status='success' AND l.curier_id=c_id
    GROUP BY com.comanda_id;

nr_comenzi NUMBER:=0;
nr_produse_total NUMBER;
livrare_curenta_id LIVRARIS.livrare_id%TYPE;
nr_curieri NUMBER;
no_curier EXCEPTION;
no_comanda EXCEPTION;

BEGIN

--verificam daca curierul exista
SELECT COUNT(*) INTO nr_curieri
FROM curieri c
WHERE c.curier_id=curier;

-- tratam cazul în care nu exista curieri cu id-ul dat
IF nr_curieri=0 THEN
    RAISE no_curier;
END IF;

-- apelam cursorul 1
OPEN livrari_total(curier);
LOOP
    FETCH livrari_total INTO livrare_curenta_id;
```

```
        EXIT WHEN livrari_total%NOTFOUND;
    END LOOP;

    IF livrari_total%rowcount=0 THEN      -- verificare daca are comenzi sau nu
-> exceptie
        CLOSE livrari_total;
        RAISE no_comanda;
    END IF;

CLOSE livrari_total;

-- apelam cursorul 2
FOR i IN comenzi_procesate_succes(curier) LOOP

    SELECT com.nr_produse INTO nr_produse_total -- in nr_produse_total vom
avea nr de produse din comanda plasata
    FROM comenzi com
    WHERE com.comanda_id=i.comanda_id;

    IF nr_produse_total=i.nr_produse THEN      -- verificam daca comanda a fost
procesata integral
        nr_comenzi:=nr_comenzi+1;
    END IF;
END LOOP;

RETURN nr_comenzi;

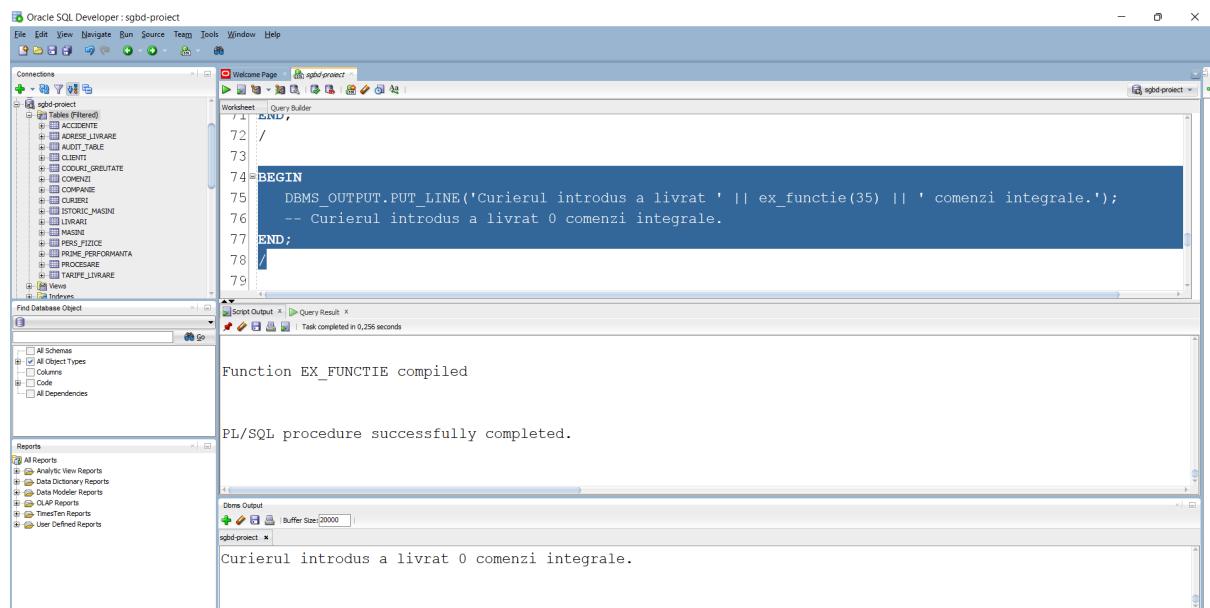
-- definim exceptiile
EXCEPTION
    WHEN no_curier THEN
        RAISE_APPLICATION_ERROR(-20017,'Niciun curier inregistrat cu id-ul
dat!');
    WHEN no_comanda THEN
        DBMS_OUTPUT.PUT_LINE('Curierul nu are deloc comenzi');
        RETURN 0;
END;
/

BEGIN
    DBMS_OUTPUT.PUT_LINE('Curierul introdus a livrat ' || ex_functie(35) || '
comenzi integrale.');
    -- Curierul introdus a livrat 0 comenzi integrale.
END;
/


BEGIN
    DBMS_OUTPUT.PUT_LINE('Curierul introdus a livrat ' || ex_functie(30) || '
comenzi integrale.');
    -- Niciun curier inregistrat cu id-ul dat!
END;
/
```

```
BEGIN
    DBMS_OUTPUT.PUT_LINE('Curierul introdus a livrat ' || ex_functie(34) || '
comenzi integrale.');
    -- Curierul introdus a livrat 1 comenzi integrale.
END;
/
```

```
BEGIN
    DBMS_OUTPUT.PUT_LINE('Curierul introdus a livrat ' || ex_functie(33) || '
comenzi integrale.');
    --Curierul nu are deloc comenzi!
    --Curierul introdus a livrat 0 comenzi integrale.
END;
/
```



The screenshot shows the Oracle SQL Developer interface with a project named "sgbd-project". In the "Worksheet" tab, there is a PL/SQL block:

```
80
81 BEGIN
82   DBMS_OUTPUT.PUT_LINE('Curierul introdus a livrat ' || ex_functie(30) || ' comenzi integrale.');
83   -- Niciun curier inregistrat cu id-ul dat!
84 END;
85 /
86
87
88 BEGIN
```

The "Script Output" tab shows the error message:

```
Error starting at line : 81 in command -
BEGIN
  DBMS_OUTPUT.PUT_LINE('Curierul introdus a livrat ' || ex_functie(30) || ' comenzi integrale.');
  -- Niciun curier inregistrat cu id-ul dat!
END;
Error report -
ORA-20017: Niciun curier inregistrat cu id-ul dat!
ORA-06512: la "UTILIZATOR.EX_FUNCTIE", linia 67
```

The screenshot shows the Oracle SQL Developer interface with the same project "sgbd-project". In the "Worksheet" tab, the PL/SQL block has been modified:

```
86
87
88 BEGIN
89   DBMS_OUTPUT.PUT_LINE('Curierul introdus a livrat ' || ex_functie(34) || ' comenzi integrale.');
90   -- Curierul introdus a livrat 1 comenzi integrale.
91 END;
92 /
93
94
95 -- BEGIN
```

The "Script Output" tab shows the successful execution message:

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

The "Query Result" tab shows the output of the DBMS\_OUTPUT.PUT\_LINE statement:

```
Curierul introdus a livrat 1 comenzi integrale.
```

The screenshot shows the Oracle SQL Developer interface. The central workspace displays a PL/SQL procedure:

```
92: /
93:
94:
95: BEGIN
96:   DBMS_OUTPUT.PUT_LINE('Curierul introdus a livrat ' || ex_functie(33) || ' comenzi integrale.');
97:   --Curierul nu are deloc comenzi!
98:   --Curierul introdus a livrat 0 comenzi integrale.
99: END;
100: /
101:
102:
```

The procedure outputs the message "Curierul nu are deloc comenzi!" and "Curierul introdus a livrat 0 comenzi integrale.".

The bottom pane shows the results of the execution:

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

Logs output window shows:

```
Curierul nu are deloc comenzi!
Curierul introdus a livrat 0 comenzi integrale.
```

9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate exceptiile care pot apărea, inclusiv exceptiile NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

- **Periodic se acorda statutul de 'client premium' celor care au plasat comenzi în valoare de minim o anumită sumă, cu minim o comandă în ultima luna. În vederea acordării acestui statut, verificati daca datele primite sunt valide(codul de identificare al clientului este valid, valoarea minima a comenzi date este pozitiva) si, în caz afirmativ, afisati id-ul, numele, prenumele și valoarea lor totală.**

```
CREATE OR REPLACE PROCEDURE proc_ex9 (client_nume PERS_FIZICE.nume%TYPE,  
valoare_totala FLOAT) IS  
  
c_id clienti.client_id%TYPE;  
c_prenume pers_fizice.prenume%TYPE;  
c_nume pers_fizice.nume%TYPE;  
email clienti.email%TYPE;  
data_co comenzi.data_comanda%TYPE;  
pret_total FLOAT;  
  
client_invalid EXCEPTION;  
valoare_invalida EXCEPTION;  
  
BEGIN  
    IF valoare_totala <= 0 THEN  
        RAISE valoare_invalida; -- tratam cazul in care valoarea  
        introdusa pentru suma este negativa sau =0  
    END IF;  
  
    SELECT c.client_id, pf.prenume, pf.nume, c.email, max(co.data_comanda),  
    SUM(co.pret_colet + tl.tarif + round((co.cod_greutate *  
    co.pret_colet)/100,2))  
    INTO c_id, c_prenume, c_nume, email, data_co, pret_total  
    FROM pers_fizice pf JOIN clienti c ON (c.client_id = pf.client_id)  
        JOIN comenzi co ON (co.client_id = c.client_id)  
        JOIN coduri_greutate cg ON (cg.cod_greutate =  
        co.cod_greutate)  
        JOIN tarife_livrare tl ON (tl.cod_identificare_tarif  
        = co.cod_identificare_tarif)  
    WHERE LOWER(pf.nume) = LOWER(client_nume)  
    GROUP BY c.client_id, pf.prenume, pf.nume, c.email;  
  
    IF pret_total < valoare_totala OR months_between(sysdate,data_co) <= 1  
    THEN -- tratam cazul in care clientul nu corespunde criteriilor  
        RAISE client_invalid;  
    END IF;
```

```
DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul ' || c_id || '(' || c_nume || '
|| c_prenume || '), email: ' || email
|| ' a efectuat comenzi cu o valoare totala de ' || 
pret_total || ' RON.');

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000,'Clientul nu exista in baza de
date!');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20001,'Exista mai multi clienti cu acest
id!');
    WHEN client_invalid THEN
        DBMS_OUTPUT.PUT_LINE('Clientul este invalid(nu corespunde criteriilor
de acordare a statutului de client premium)');
    WHEN valoare_invalida THEN
        RAISE_APPLICATION_ERROR(-20002,'Valoarea este invalida. Se astepta un
numar pozitiv!');
END;
/

BEGIN
    proc_ex9('Iordache', 0); --Valoarea este invalida. Se astepta un numar
pozitiv!
END;
/

BEGIN
    proc_ex9('Iordache', 100); -- Clientul cu id-ul 10006(Iordache
Beatrice), email: iordache_bety@yahoo.com a efectuat comenzi cu o valoare
totala de 1258,3 RON.
END;
/

BEGIN
    proc_ex9('Corneanu', 100); -- Clientul nu exista in baza de date!
END;
/

BEGIN
    proc_ex9('Ionescu', 500); -- Clientul este invalid(nu corespunde
criteriilor de acordare a statutului de client premium)
END;
/

BEGIN
    proc_ex9('Popescu', 100); --Exista mai multi clienti cu acest id!
END;
/
```

```

37      WHEN TOO_MANY_ROWS THEN
38          RAISE_APPLICATION_ERROR(-20001,'Există mai mulți clienți cu acest id!');
39      WHEN client_invalid THEN
40          DBMS_OUTPUT.PUT_LINE('Cliențul este invalid(nu corespunde criteriilor de acordare a statutului de');
41      WHEN valoare_invalida THEN
42          RAISE_APPLICATION_ERROR(-20002,'Valoarea este invalidă. Se aștepta un număr pozitiv!');
43  END;
44/
45
46 BEGIN
47     proc_ex9('Iordache', 0); --Valoarea este invalidă. Se aștepta un număr pozitiv!
48 END;
49/

```

Script Output | Query Result | Task completed in 0,089 seconds

```

BEGIN
proc_ex9('Iordache', 0); --Valoarea este invalidă. Se aștepta un număr pozitiv!
END;
Error report -
ORA-20000: Valoarea este invalidă. Se aștepta un număr pozitiv!
ORA-06512: la "UTILIZATOR.PROC_EX9", linia 42
ORA-06512: la linia 2

```

Items Output

```

46 BEGIN
47     proc_ex9('Iordache', 0); --Valoarea este invalidă. Se aștepta un număr pozitiv!
48 END;
49/
50
51 BEGIN
52     proc_ex9('Iordache', 100); -- Clientul cu id-ul 10006(Iordache Beatrice), email: iordache_betty@yahoo.com a efectuat comenzi cu o valoare totală
53 END;
54/
55

```

Script Output | Query Result | Task completed in 0,088 seconds

PL/SQL procedure successfully completed.

Items Output | Buffer Size:20000 | sgbd-project.x

Clientul cu id-ul 10006(Iordache Beatrice), email: iordache\_betty@yahoo.com a efectuat comenzi cu o valoare totală

The screenshot shows the Oracle SQL Developer interface with a project named "sgbd-project". In the "Worksheet" tab, a PL/SQL block is being run:

```

52     proc_ex9('Iordache', 100); -- Clientul cu id-ul 10006(Iordache Beatrice), email: iordache_betty@yahoo
53 END;
54 /
55
56 BEGIN
57     proc_ex9('Corneanu', 100); -- Clientul nu exista in baza de date!
58 END;
59 /
60
61 BEGIN
62     proc_ex9('Ionescu', 500); -- Clientul este invalid(nu corespunde criteriilor de acordare a statutului)
63 END;
64 /
65

```

The "Script Output" panel shows the following errors:

```

ORA-20000: Clientul nu exista in baza de date!
ORA-06512: la "UTILIZATOR.PROC_EX9", linia 36
ORA-06512: la linia 2
20000. 00000 - "%s"
*Cause: The stored procedure 'raise_application_error'
          was called which causes this error to be generated.

```

The screenshot shows the Oracle SQL Developer interface with the same project "sgbd-project". The "Worksheet" tab contains the same PL/SQL block as the previous screenshot, but it has been successfully executed.

The "Script Output" panel displays the message:

```

PL/SQL procedure successfully completed.

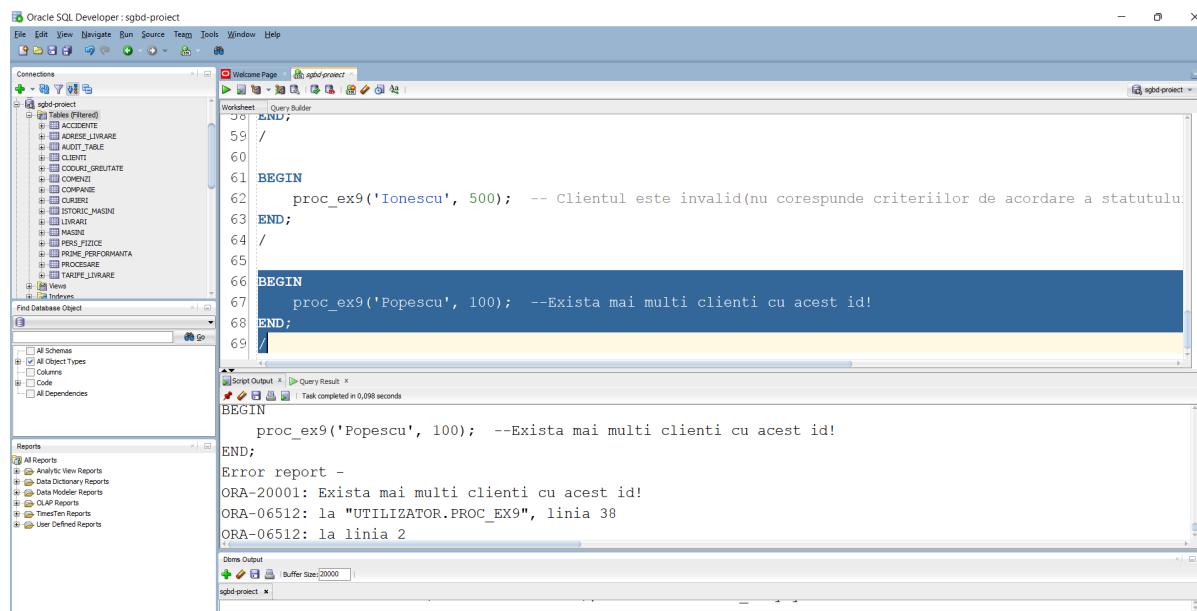
```

The "Query Result" panel shows the output:

```

Clientul este invalid(nu corespunde criteriilor de acordare a statutului de client premium)

```



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

**Deoarece compania de curierat a avut încasări mai mici în ultima jumătate de an, a decis că reduceri de buget. În urma acestei decizii, s-a hotarat ca numărul de prime oferite să nu depasească 4, dar să existe cel puțin una pentru a stimula angajații periodic. Creați un trigger pentru a soluționa acest aspect.**

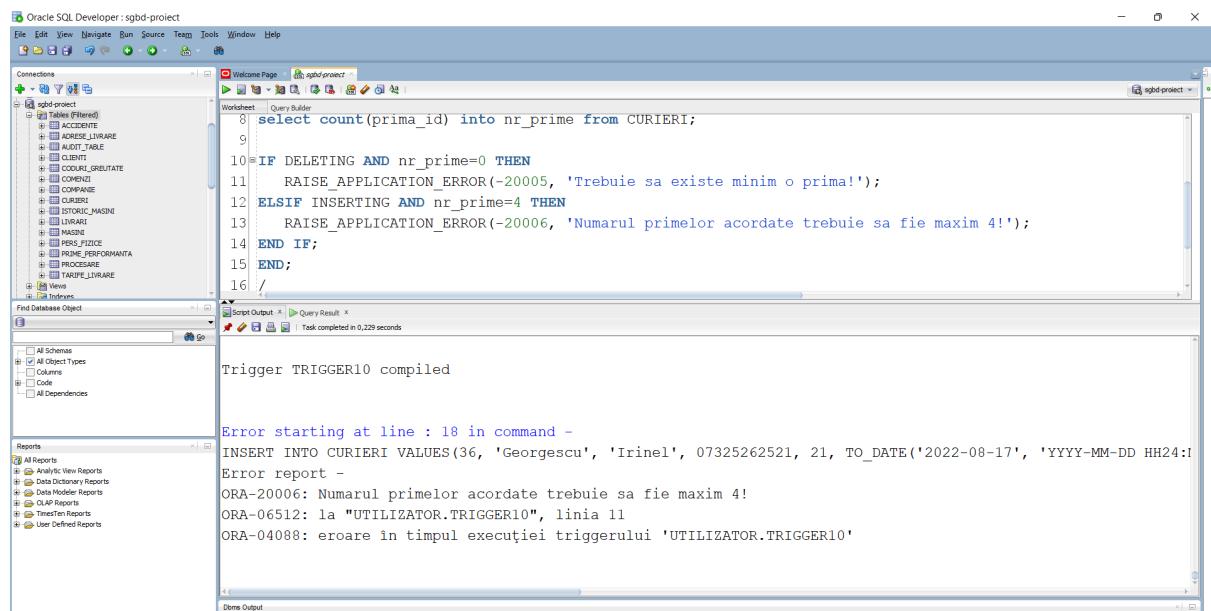
```
CREATE OR REPLACE TRIGGER trigger10
    BEFORE INSERT OR DELETE ON CURIERI
DECLARE
    nr_prime NUMBER;

BEGIN

    select count(prima_id) into nr_prime from CURIERI;

    IF DELETING AND nr_prime=0 THEN
        RAISE_APPLICATION_ERROR(-20005, 'Trebuie să existe minim o prima!');
    ELSIF INSERTING AND nr_prime>4 THEN
        RAISE_APPLICATION_ERROR(-20006, 'Numarul primelor acordate trebuie să fie maxim 4!');
    END IF;
END;
/

INSERT INTO CURIERI VALUES(36, 'Georgescu', 'Irinel',
07325262521, 21, TO_DATE('2022-08-17', 'YYYY-MM-DD
HH24:MI:SS'), 3200, 2, 220, TO_DATE('2022-08-17', 'YYYY-MM-DD
HH24:MI:SS'));
```



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

- **Să se afișeze o eroare atunci când în urma modificării salariului sau bonusului pentru un curier care lucrează de mai puțin de 5 ani, acesta are un venit total(salariu\_brut + salariu\_brut\*procent/100+prima\_ocazionala) mai mare decat cel mediu.**

```
CREATE OR REPLACE TRIGGER trigger11
    BEFORE UPDATE OF SALARIU_BRUT, PRIMA_ID, PRIMA_OCAZIONALA ON CURIERI
    FOR EACH ROW
DECLARE
    PRAGMA autonomous_transaction; --pt mutating table
    d_ang DATE;
    salariu_mediul FLOAT;
    procent_nou PRIME_PERFORMANTA.PROCENT%TYPE;
    venit FLOAT;
    exceptie EXCEPTION;

BEGIN
    SELECT data_angajare INTO d_ang FROM curieri WHERE CURIER_ID =
    :NEW.curier_id;

    SELECT
        AVG(c.SALARIU_BRUT+(nvl(p.procent,0)*c.SALARIU_BRUT)/100+c.PRIMA_OCAZIONALA)
    INTO salariu_mediul
    FROM CURIERI c
    LEFT JOIN PRIME_PERFORMANTA p USING(prima_id);

    IF :NEW.prima_id IS NULL THEN
        procent_nou:=0;
        venit:=:NEW.salariu_brut + :NEW.prima_ocazionala;
    ELSE
        SELECT PROCENT INTO procent_nou
        FROM PRIME_PERFORMANTA
        WHERE prima_id =:NEW.prima_id;

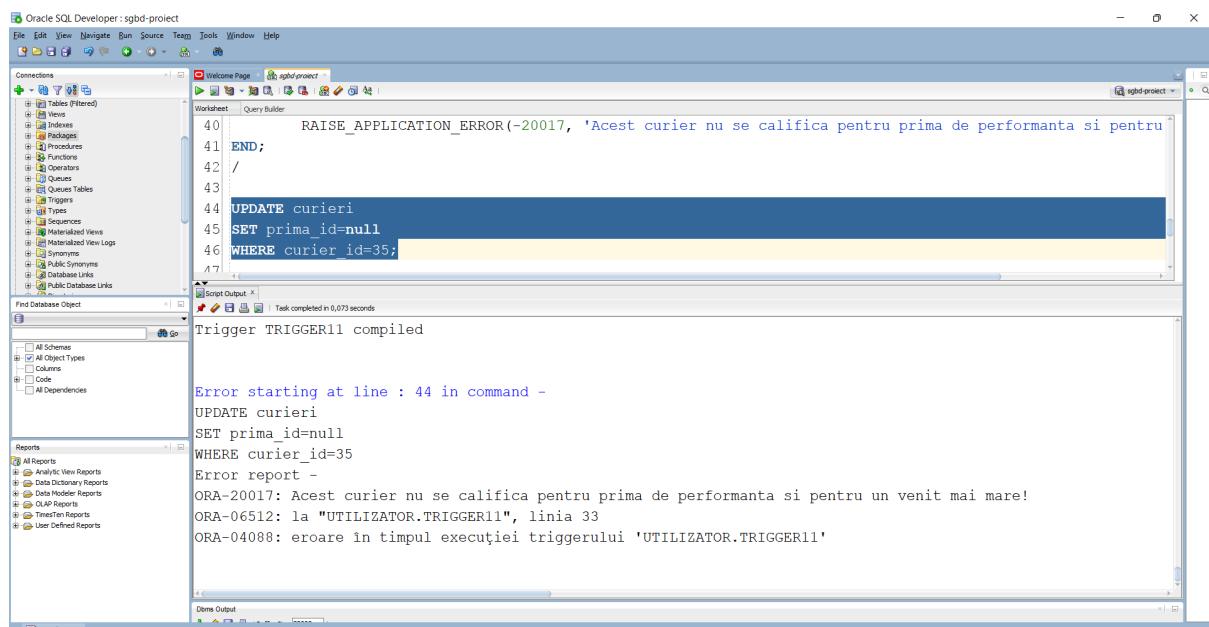
        venit:=:NEW.salariu_brut+ (:NEW.salariu_brut*procent_nou)/100
        + :NEW.prima_ocazionala;
    END IF;
    --verificam daca curierul este valid pentru prima
    IF venit > salariu_mediul OR months_between(sysdate, d_ang)< 60
    THEN
        RAISE exceptie;
    END IF;
EXCEPTION
    WHEN exceptie THEN
        RAISE_APPLICATION_ERROR(-20017, 'Acest curier nu se califica pentru
        prima de performanta si pentru un venit mai mare!');
END;
/
```

```
UPDATE curieri
SET prima_id=null
WHERE curier_id=35; -- Acest curier nu se califica pentru prima de
performanta si pentru un venit mai mare!
```

```
UPDATE curieri
SET PRIMA_ID=3
WHERE curier_id=33; -- Acest curier nu se califica pentru prima de
performanta si pentru un venit mai mare!
```

```
rollback;
```

```
UPDATE curieri
SET prima_id=null
WHERE curier_id=100; -- => 0 rows updated
```



12. Definiți un trigger de tip LDD. Declanșați trigger-ul.  
**Înregistrați toate operațiile LDD efectuate asupra schemei folosind un tabel auxiliar *audit\_table*.**

```

CREATE TABLE audit_table
(
    event_id NUMBER PRIMARY KEY,
    username VARCHAR2(100) NOT NULL,
    date_event DATE NOT NULL,
    event VARCHAR2(50) NOT NULL,
    object_name VARCHAR2(100) NOT NULL
);

CREATE SEQUENCE seq_audit_id MINVALUE 1 MAXVALUE 9999999 INCREMENT BY 1
START WITH 1 NOCACHE ;

--procedura care insereaza in tabelul audit_table
CREATE OR REPLACE PROCEDURE proc_add(event_id NUMBER, username VARCHAR2,
date_event DATE, event VARCHAR2, object_name VARCHAR2) IS
BEGIN
    INSERT INTO audit_table
VALUES(event_id,username,date_event,event,object_name);
END;
/

```

```

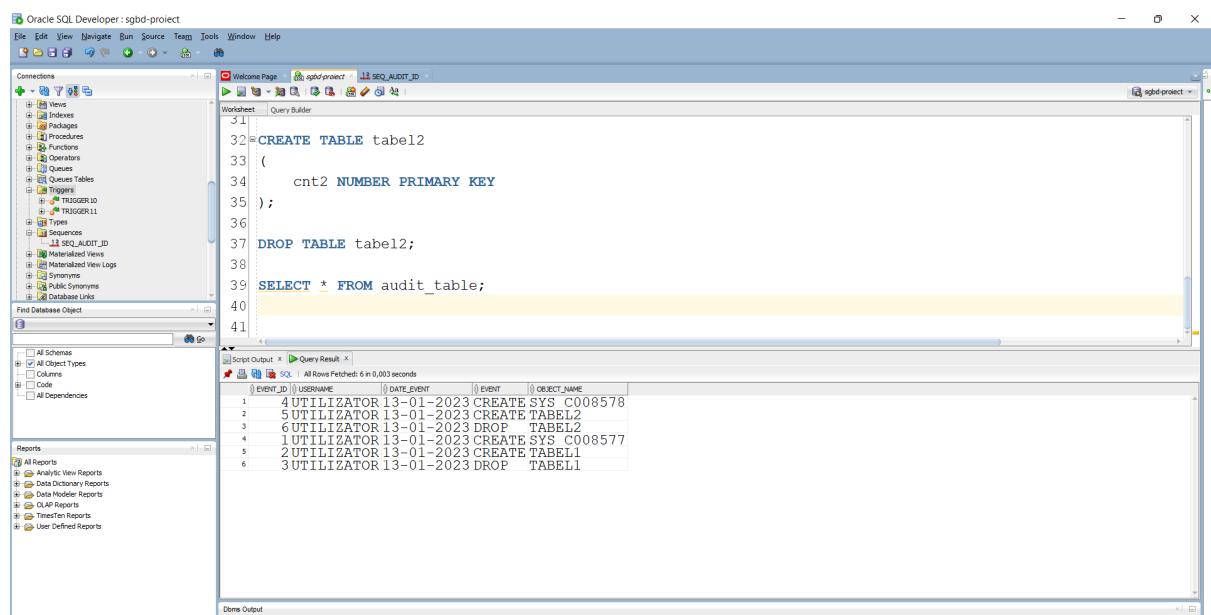
CREATE OR REPLACE TRIGGER trigger12
    AFTER DROP OR ALTER OR CREATE ON SCHEMA
BEGIN
    proc_add(seq_audit_id.NEXTVAL, user, sysdate, ora_sysevent,
SYS.dictionary_obj_name);
END;
/

```

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

- Connections:** Shows various database objects like Views, Indexes, Procedures, Triggers, etc.
- Script Output:** Displays the execution results of the SQL statements:
  - Sequence AUDIT\_ID created.
  - Procedure PROC\_ADD compiled
  - Trigger TRIGGER12 compiled
- Query Result:** Shows the output of the SELECT statement: "Sequence AUDIT\_ID created."
- Task completed in 0,107 seconds:** Confirmation of the successful execution.

```
SELECT * FROM audit_table;
```



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

```
CREATE OR REPLACE PACKAGE ex_package IS

PROCEDURE ex_colectii; --6
PROCEDURE ex_cursoare; --7
FUNCTION ex_functie(curier curieri.curier_id%TYPE) RETURN NUMBER; --8
PROCEDURE proc_ex9 (client_nume PERS_FIZICE.nume%TYPE, valoare_totala
FLOAT); --9
PROCEDURE proc_add(event_id NUMBER, username VARCHAR2, date_event DATE, event
VARCHAR2, object_name VARCHAR2);
END ex_package;
/

CREATE OR REPLACE PACKAGE BODY ex_package IS
--6
PROCEDURE ex_colectii IS
    TYPE t_judete IS TABLE OF adrese_livrare.judet%type INDEX BY PLS_INTEGER;
--index-by-table
    v_judete t_judete;
    TYPE t_nr_comenzi IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
--index-by-table
    v_nr_comenzi t_nr_comenzi;
    TYPE t_orase IS TABLE OF ADRESE_LIVRARE.oras%TYPE; --nested table
    TYPE t_total_orase IS TABLE OF t_orase; --nested table
    v_orase t_total_orase:=t_total_orase();
    nr_judete NUMBER;
    nr_orase NUMBER;

BEGIN
    SELECT DISTINCT ad.judet, COUNT(*)
    BULK COLLECT INTO v_judete, v_nr_comenzi
    FROM comenzi c JOIN ADRESE_LIVRARE ad ON c.adresa_livrare_id=ad.ADRESA_id
    GROUP BY ad.judet;

    nr_judete:=v_judete.COUNT;

    FOR i IN 1..nr_judete LOOP -- pentru fiecare judet
        v_orase.EXTEND;

        SELECT DISTINCT ad.oras
        BULK COLLECT INTO v_orase(i)
        FROM comenzi c JOIN ADRESE_LIVRARE ad ON
        c.adresa_livrare_id=ad.adresa_id
        WHERE ad.judet=v_judete(i);
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Detalii comenzi pe județe: ');

    FOR i IN 1..nr_judete LOOP -- pentru fiecare judet
        DBMS_OUTPUT.PUT(v_judete(i) || ' are ' || v_nr_comenzi(i));
        IF v_nr_comenzi(i)=1 THEN

```

```
        DBMS_OUTPUT.PUT(' comanda in ');
ELSE
    DBMS_OUTPUT.PUT(' comenzi in ');
END IF;

        DBMS_OUTPUT.PUT(v_orase(i).COUNT);
IF v_orase(i).COUNT=1 THEN
    DBMS_OUTPUT.PUT(' oras(');
ELSE
    DBMS_OUTPUT.PUT(' orase(');
END IF;

nr_orase:=v_orase(i).COUNT;
FOR j IN 1..nr_orase-1 LOOP
    DBMS_OUTPUT.PUT( v_orase(i)(j)|| ', ');
END LOOP;
DBMS_OUTPUT.PUT( v_orase(i)(nr_orase)|| ')');

        DBMS_OUTPUT.NEW_LINE;
END LOOP;
END ex_colectii;

--7
PROCEDURE ex_cursoare IS

an number(4) := &p_an;

-- cursor parametrizat
CURSOR c1_ex_cursoare(an NUMBER) IS
    SELECT c.curier_id curier_id, c.prenume prenume, c.nume nume,
a.data_accident data_accident, a.cost_daune cost_daune, m.masina_id
masina_id,
m.marca marca, m.model model, c.data_angajare data_angajare,
istm.data_preluare_masina data_preluare_masina, istm.data_predare_masina,
c.data_preluare_masina data_preluare_masina2
    FROM curieri c JOIN accidente a ON c.curier_id=a.curier_id
    JOIN masini m ON a.masina_id=m.masina_id
    LEFT JOIN istoric_masini istm ON
(a.masina_id=istm.masina_id AND a.curier_id=istm.curier_id
    AND
a.data_accident<=istm.DATA_PREDARE_MASINA AND
a.DATA_ACCIDENT>=istm.DATA_PRELUARE_MASINA )
    WHERE EXTRACT(YEAR FROM a.data_accident) = an;

curenta VARCHAR2(50);
top number(1) := 0;

BEGIN
DBMS_OUTPUT.PUT_LINE('Top 3 accidente 2022 conform pagubelor');
--ciclul cursor cu subcererii
FOR i IN (SELECT c.curier_id id, c.prenume p, c.nume n, a.data_accident da,
a.cost_daune cd, c.salariu_brut sb
```

```

        FROM curieri c JOIN accidente a ON c.curier_id=a.curier_id
        ORDER BY a.cost_daune DESC)
    LOOP
        DBMS_OUTPUT.PUT_LINE('Data accident: ' || i.da || '; Sofer
responsabil:' || i.id || '; Daune materiale:' || i.cd);
        top := top+1;
        EXIT WHEN top = 3;
    END LOOP;

DBMS_OUTPUT.PUT_LINE('Accidente inregistrate in anul ' || an);
FOR i IN c1_ex_cursoare(an) LOOP --ciclul cursor

    IF i.data_predare_masina < i.data_accident THEN
        curenta:='masina curenta a curierului ';
    ELSE
        curenta:='masina anterioara a curierului ';
    END IF;
    DBMS_OUTPUT.PUT_LINE('Masina cu numarul de identificare(id) ' ||
i.masina_id || '(' || i.marca || ' ' || i.model || ',' || curenta || i.nume
|| ' ' || i.prenume || ', avand id-ul ' || i.curier_id || ', a fost implicata
intr-un accident in luna ' || extract(month from i.data_accident) || '.');
END LOOP;

END ex_cursoare;

--8
FUNCTION ex_functie(curier curieri.curier_id%TYPE) RETURN NUMBER IS

CURSOR livrari_total(c_id curieri.curier_id%TYPE) IS
-- cursor 1 pentru livrarile efectuate de un curier c_id
    SELECT livrare_id
    FROM livrari
    WHERE curier_id=c_id;

CURSOR comenzi_procesate_succes(c_id curieri.curier_id%TYPE) IS
-- cursor 2 pentru comenziile procesate 'success'
    SELECT com.comanda_id comanda_id, SUM(p.NR_PRODUSE) nr_produse
    FROM comenzi com  JOIN procesare p ON com.comanda_id=p.comanda_id
        JOIN livrari l ON l.LIVRAR_ID=p.LIVRAR_ID
    WHERE p.status='success' AND l.curier_id=c_id
    GROUP BY com.comanda_id;

nr_comenzi NUMBER:=0;
nr_produse_total NUMBER;
livrare_curenta_id LIVRAR.livrare_id%TYPE;
nr_curieri NUMBER;
no_curier EXCEPTION;
no_comanda EXCEPTION;

BEGIN
    --verificam daca curierul exista

```

```
SELECT COUNT(*) INTO nr_curieri
FROM curieri c
WHERE c.curier_id=curier;

-- cazul in care nu exista curieri cu id-ul dat
IF nr_curieri=0 THEN
    RAISE no_curier;
END IF;

-- apelam cursorul 1
OPEN livrari_total(curier);
LOOP
    FETCH livrari_total INTO livrare_curenta_id;
    EXIT WHEN livrari_total%NOTFOUND;
END LOOP;

IF livrari_total%rowcount=0 THEN      -- verificare daca are comenzi sau nu
-> exceptie
    CLOSE livrari_total;
    RAISE no_comanda;
END IF;

CLOSE livrari_total;

-- apelam cursorul 2
FOR i IN comenzi_procesate_succes(curier) LOOP

    SELECT com.nr_produse INTO nr_produse_total -- in nr_produse_total vom
avea nr de produse din comanda plasata
    FROM comenzi com
    WHERE com.comanda_id=i.comanda_id;

    IF nr_produse_total=i.nr_produse THEN      -- verificam daca comanda a fost
procesata integral
        nr_comenzi:=nr_comenzi+1;
    END IF;
END LOOP;

RETURN nr_comenzi;

-- definim exceptiile
EXCEPTION
    WHEN no_curier THEN
        RAISE_APPLICATION_ERROR(-20017,'Niciun curier inregistrat cu id-ul
dat!');
    WHEN no_comanda THEN
        DBMS_OUTPUT.PUT_LINE('Curierul nu are deloc comenzi!');
        RETURN 0;
END ex_functie;

--9
PROCEDURE proc_ex9 (client_nume PERS_FIZICE.nume%TYPE, valoare_totala FLOAT)
IS
```

```
c_id clienti.client_id%TYPE;
c_prenume pers_fizice.prenume%TYPE;
c_nume pers_fizice.nume%TYPE;
email clienti.email%TYPE;
data_co comenzi.data_comanda%TYPE;
pret_total FLOAT;

client_invalid EXCEPTION;
valoare_invalida EXCEPTION;

BEGIN
    IF valoare_totala <= 0 THEN
        RAISE valoare_invalida;          -- tratam cazul in care valoarea
introdusa pentru suma este negativa sau =0
    END IF;

    SELECT c.client_id, pf.prenume, pf.nume, c.email, max(co.data_comanda),
SUM(co.pret_colet + tl.tarif + round((co.cod_greutate *
co.pret_colet)/100,2))
    INTO c_id, c_prenume, c_nume, email, data_co, pret_total
    FROM pers_fizice pf JOIN clienti c ON (c.client_id = pf.client_id)
                      JOIN comenzi co ON (co.client_id = c.client_id)
                      JOIN coduri_greutate cg ON (cg.cod_greutate =
co.cod_greutate)
                      JOIN tarife_livrare tl ON (tl.cod_identificare_tarif
= co.cod_identificare_tarif)
    WHERE LOWER(pf.nume) = LOWER(client_nume)
    GROUP BY c.client_id, pf.prenume, pf.nume, c.email;

    IF pret_total < valoare_totala OR months_between(sysdate,data_co) <= 1
THEN      -- tratam cazul in care clientul nu corespunde criteriilor
        RAISE client_invalid;
    END IF;

    DBMS_OUTPUT.PUT_LINE('Clientul cu id-ul ' || c_id || '(' || c_nume || '
|| c_prenume || '), email: ' || email
|| ' a efectuat comenzi cu o valoare totala de ' || pret_total || ' RON.');
END;
```

EXCEPTION

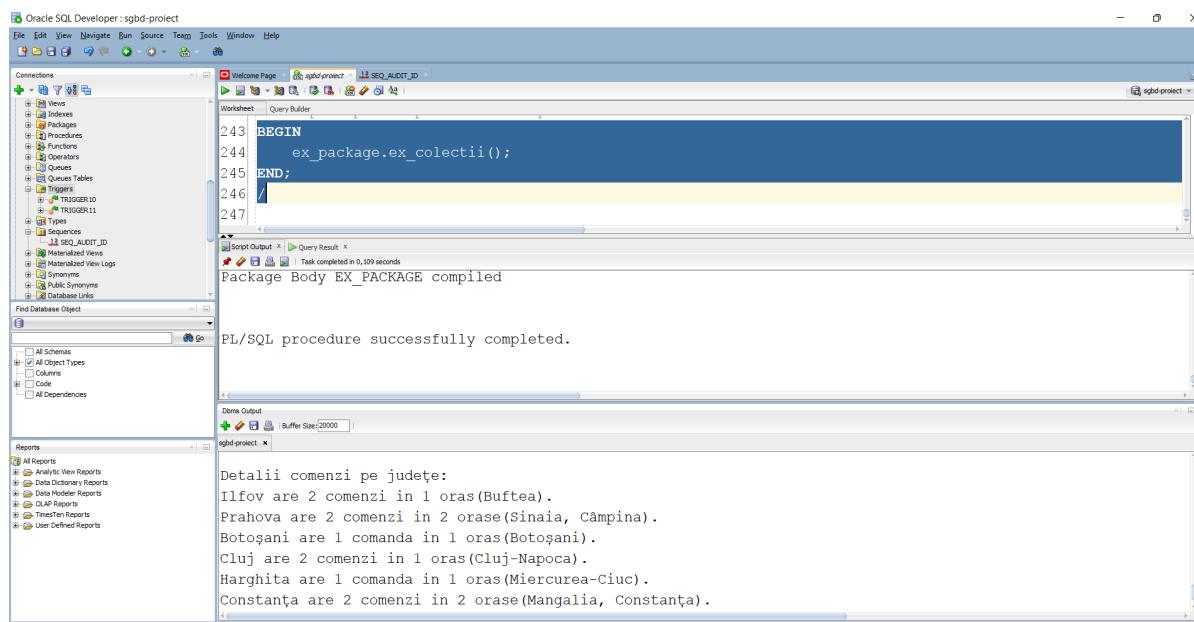
```
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000,'Clientul nu exista in baza de
date!');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20001,'Exista mai multi clienti cu acest
id!');
    WHEN client_invalid THEN
        DBMS_OUTPUT.PUT_LINE('Clientul este invalid(nu corespunde criteriilor
de acordare a statutului de client premium)');
    WHEN valoare_invalida THEN
        RAISE_APPLICATION_ERROR(-20002,'Valoarea este invalida. Se astepta un
numar pozitiv!');
```

```
END proc_ex9;
```

```
PROCEDURE proc_add(event_id NUMBER, username VARCHAR2, date_event DATE, event
VARCHAR2, object_name VARCHAR2) IS
BEGIN
    INSERT INTO audit_table
VALUES(event_id,username,date_event,event,object_name);
END proc_add;

END ex_package;
/

--exemplu apelare procedura din pachet
BEGIN
    ex_package.ex_colectii();
END;
/
```



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).

- **Verificați dacă profitul pe luna decembrie a fost mai mare decat cel din noiembrie. În caz afirmativ, mariti salariile primilor 3 curieri care au livrat cele mai multe pachete cu 15% și afișați informații despre aceștia (id, nume, număr de pachete livrate); altfel, măriți tarifele de livrare cu 20% și afișați care ar fi fost profitul pe decembrie cu aceste noi preturi.**

```
CREATE OR REPLACE PACKAGE package_ex14 IS
    TYPE curieri_obj_type IS RECORD
    (
        curier_id NUMBER,
        nr_produse NUMBER,
        nume VARCHAR2(50)
    );
    TYPE top_curieri IS TABLE OF curieri_obj_type;

    FUNCTION profit_comenzi(luna NUMBER) RETURN NUMBER;
    FUNCTION get_top_curieri RETURN top_curieri;
    PROCEDURE marire_salarii;
    PROCEDURE marire_tarife_livrare;
    PROCEDURE aplica;

END package_ex14;
/

CREATE OR REPLACE PACKAGE BODY package_ex14 IS

    FUNCTION profit_comenzi(luna NUMBER) RETURN NUMBER IS
        total_profit NUMBER;
        no_comanda EXCEPTION;
    BEGIN
        --calculam profitul pe o anumita luna
        SELECT SUM((co.pret_colet*cg.supracost_procent)/100 + tl.tarif)
        INTO total_profit
        FROM comenzi co JOIN coduri_greutate cg USING (cod_greutate)
        JOIN tarife_livrare tl USING
        (cod_identificare_tarif)
        WHERE EXTRACT(MONTH FROM (co.data_comanda))=luna;

        IF total_profit IS NULL THEN
            RAISE no_comanda;
            --tratam cazul in care nu a fost inregistrat profit pe o anumita luna
        END IF;

        RETURN total_profit;

    EXCEPTION
        WHEN no_comanda THEN
```

```
        DBMS_OUTPUT.PUT_LINE('În aceasta luna nu s-a plasat nicio
comandă!');

        RETURN 0;

END profit_comenzi;

FUNCTION get_top_curieri RETURN top_curieri IS
CURSOR c_curieri IS
        --determinam top 3 curieri
        SELECT t.curier_id, t.nume, SUM(t.nr_produse)
total_colete_livrare
        FROM
        (SELECT curier_id, nume, livrare_id, COUNT(*)
nr_produse
        FROM livrari JOIN procesare USING (livrare_id)
        JOIN curieri USING(curier_id)
        WHERE status='success'
        GROUP BY livrare_id, curier_id, nume) t
        GROUP BY t.curier_id, t.nume
        ORDER BY 3 DESC
        FETCH FIRST 3 ROWS ONLY;

j NUMBER:=0;
v_top_curieri top_curieri:=top_curieri();

BEGIN

FOR i IN c_curieri LOOP
        j:=j+1;
        v_top_curieri.EXTEND;
        v_top_curieri(j):=curieri_obj_type(i.curier_id,
i.total_colete_livrare, i.nume);
    END LOOP;

RETURN v_top_curieri;

END get_top_curieri;

--procedura care marestе salariile curierilor si afiseaza aceasta
'operatiune'
PROCEDURE marire_salarii IS
v_curieri top_curieri;
BEGIN

v_curieri:=get_top_curieri;

FOR i IN 1..v_curieri.COUNT LOOP
        UPDATE curieri SET salariu_brut=ROUND(salariu_brut*1.15)
        WHERE curier_id=v_curieri(i).curier_id;

        DBMS_OUTPUT.PUT_LINE('Curierului cu id-ul ' || v_curieri(i).curier_id
|| '(' || v_curieri(i).nume
```

```
    || ') i s-a mărit salariul cu 15%, deoarece s-a
clasat in top 3 cei mai buni curieri livând '
    || v_curieri(i).nr_produse|| ' colete.';

END LOOP;

END marire_salarii;

--procedura care mareste tarifele
PROCEDURE marire_tarife_livrare IS
BEGIN
    UPDATE tarife_livrare
    SET tarif=tarif*1.25;

END marire_tarife_livrare;

PROCEDURE aplica IS
v_profit_nov FLOAT;
v_profit_dec FLOAT;

BEGIN
    v_profit_nov:=profit_comenzi(11);
    v_profit_dec:=profit_comenzi(12);

    - conditie comparare profit => marire salarii/tarife+afisare
    IF v_profit_dec>v_profit_nov THEN
        marire_salarii;
    ELSE
        marire_tarife_livrare;
        DBMS_OUTPUT.PUT_LINE('Dupa aplicarea schimbarilor, profitul va fi
' || profit_comenzi(11) || ' in loc de'
                                || v_profit_dec || '.');
    END IF;

END aplica;

END package_ex14;
/

BEGIN
    package_ex14.aplica();
    -- (trigger11) Acest curier nu se califica pentru prima de performanta si
    pentru un venit mai mare!

    -- drop trigger trigger11;
    --Curierului cu id-ul 34( Andronic) i s-a mărit salariul cu 15%, deoarece
    s-a clasat in top 3 cei mai buni curieri livând 2 colete.
    --Curierului cu id-ul 32( Ionescu) i s-a mărit salariul cu 15%, deoarece
    s-a clasat in top 3 cei mai buni curieri livând 2 colete.
    --Curierului cu id-ul 35( Ilie ) i s-a mărit salariul cu 15%, deoarece
    s-a clasat in top 3 cei mai buni curieri livând 1 colete.
```

```
END;  
/
```

```
--Pentru:  
--      v_profit_nov:=profit_comenzi(9);  
--      v_profit_dec:=profit_comenzi(10);  
--În aceasta luna nu s-a plasat nicio comandă!  
--Acest curier nu se califica pentru prima de performanta si pentru un venit  
mai mare!
```