PROIECT SGBD

Ionescu Radu-Constantin, grupa 234

1. Descrierea modelului real, utilitatea acestuia si regulile de functionare

Acest proiect consta in implementarea unei baze de date pentru a fi folosita de o multinationala in domeniul retail-ului. Astfel, baza de date permite eficientizarea proceselor zilnice din cadrul companiei si gestiunea eficienta a magazinelor, angajatilor, vanzarilor, produselor si furnizorilor.

Nucleul bazei de date se afla in legaturile dintre magazine, produse si furnizori, oferind o mai buna evidenta a acestora si posibilitati de interogari rapide in legatura cu domenii relevante: venitul intr-o perioada de timp, numarul angajatilor per magazin/locatie/departament/oras/tara, evidenta vechimii angajatilor, analiza rentabilitatii unui magazin individual si calculul profitului companiei intr-un interval de timp, etc.

Regulile de funcționare:

-furnizorii ofera multiple produse mai multor magazine

-fiecare produs se incadreaza intr-o singura categorie de produse si se poate comercializa en-gross sau en-detail

-intr-o tara se afla mai multe orase cu mai multe adrese la care se afla magazine

-un angajat apartine de un singur departament si un singur magazin

-companiile de mentenanta ofera servicii de securitate si curatenie magazinelor pentru care lucreaza

-toti angajatii unui departament au acelasi salariu de baza, peste care se adauga un eventual comision individual

Constrangeri

Pentru a putea fi operational, modelul respecta urmatoarele:

-fiecarei adrese ii corespunde exact un magazin si viceversa

-un angajat lucreaza la un singur departament si la un singur magazin

-un bon se realizeaza la un singur magazin

-un produs are un unic furnizor si o singura categorie din care face parte

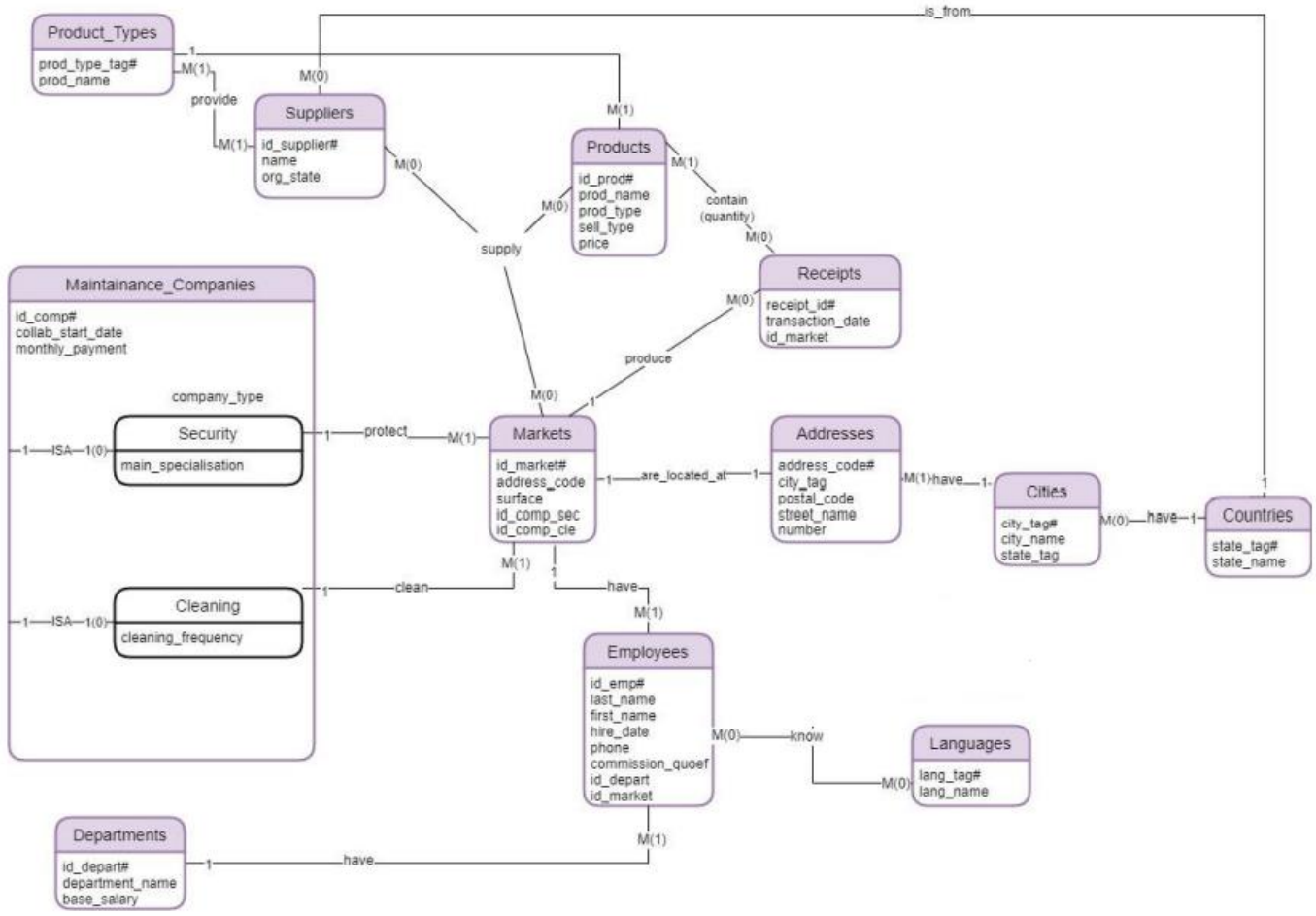
-un produs nu poate costa mai mult de 999.9 euro

-se pot cumpara cel mult 99 de produse de acelasi fel pe un singur bon sau 99 de kilograme en-gros

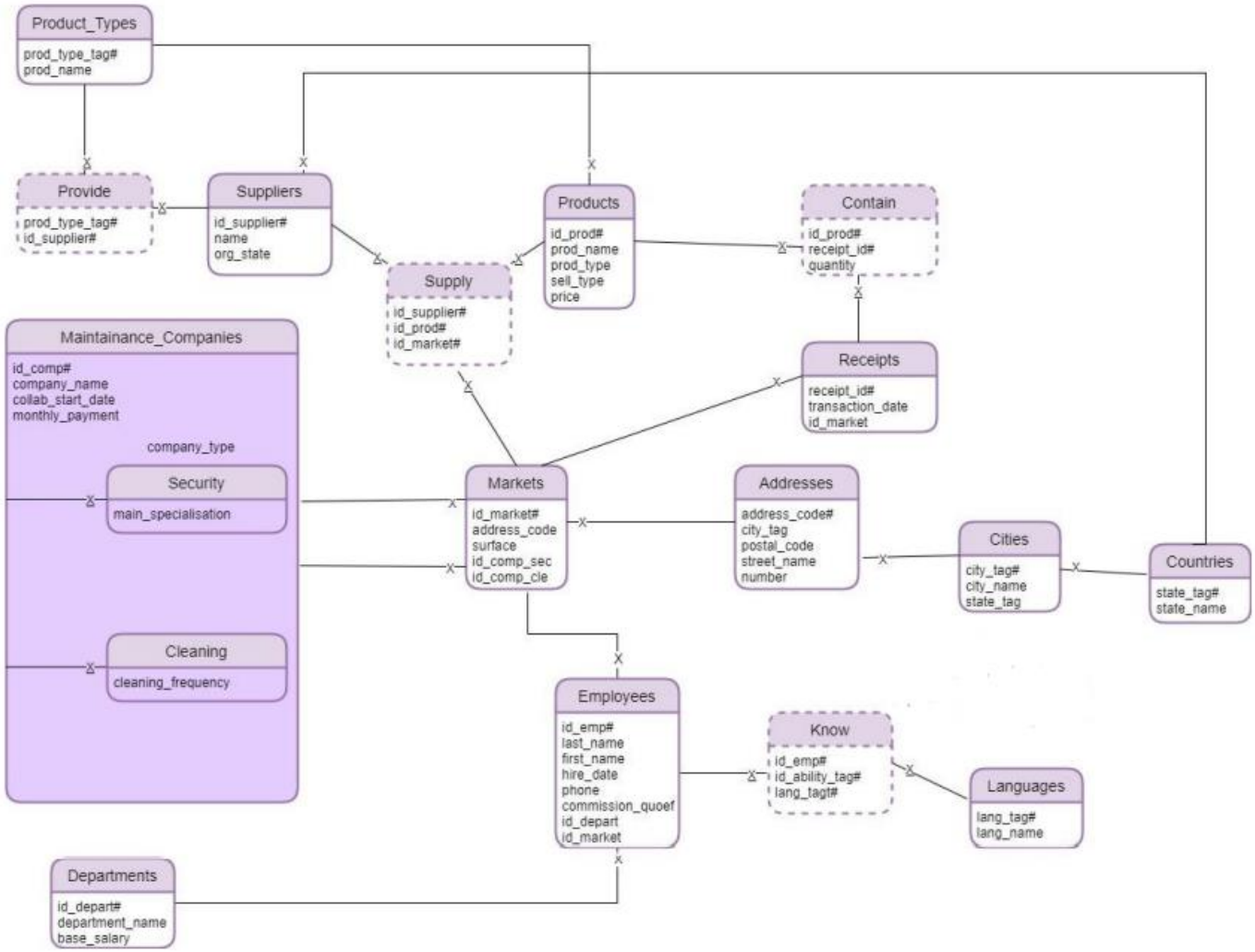
-un furnizor apartine unui singur stat deorece se presupune ca nu aducem produse prin intermediari, ci se procura de la producatori locali)

-un magazin trebuie sa aiba exact o firma de curatenie si o firma de securitate -o adresa se afla intr-un singur oras si un oras se afla intr-o singura tara

2. Diagrama ERD



3. Diagrama conceptuala



4. Definirea tabeleleor in Oracle

```
create table countries(
state_tag varchar(3) primary key,
state_name varchar(30));

create table cities(
city_tag varchar(3) primary key,
city_name varchar(20),
state_tag varchar(30),
```

```
constraint cities_fk foreign key(state_tag) references countries(state_tag));

create table addresses(
address_code number(4) primary key,
city_tag varchar(3) not null,
postal_code varchar(15),
street_name varchar(30) not null,
number_s number(3) not null,
constraint addresses_fk foreign key (city_tag) references cities(city_tag));

create sequence addresses_seq
increment by 20
start with 20
maxvalue 10000
nocycle;

create table departments(
id_depart number(3) primary key,
departmentg_name varchar(20) not null,
base_salary number(5) not null);

create sequence departments_seq
increment by 10
start with 20
maxvalue 1000
nocycle;
```

```
create table maintainance_companies(
id_comp number(3) primary key,
company_name varchar(15),
collab_start_date date,
monthly_payment number(5) not null,
company_type varchar(12) not null,
main_specialisation varchar(20),
cleaning_frequency number(2));

create sequence maintainance_seq
increment by 2
start with 10
maxvalue 1000
nocycle;

create table markets(
id_market number(4) primary key,
address_code number(4) not null,
surface number(4) not null,
id_comp_sec number(3),
id_comp_cle number(3),
constraint markets_fk foreign key(address_code) references addresses(address_code),
constraint sec_fk foreign key(id_comp_sec) references maintainance_companies(id_comp),
constraint cle_fk foreign key(id_comp_cle) references maintainance_companies(id_comp));

create sequence markets_seq
increment by 5
start with 10
```

```
maxvalue 1000
nocycle;

create table employees(
id_emp number(4) primary key,
last_name varchar(20) not null,
first_name varchar(20) not null,
hire_date date,
phone varchar(10) unique,
commission_quoef number(3,2),
id_depart number(3),
id_market number(4),
constraint dep_fk foreign key(id_depart) references departments(id_depart),
constraint market_fk foreign key(id_market) references markets(id_market));

create sequence emp_seq
increment by 1
start with 1
maxvalue 20000
nocycle;

create table languages(
lang_tag varchar(3) primary key,
lang_name varchar(20) not null);

create table product_types(
prod_type_tag varchar(3) primary key,
prod_name varchar(20) not null);
```

```
create table suppliers(
id_supplier  number(3) primary key,
name_s varchar(20) not null,
org_state varchar(3),
constraint sup_fk foreign key(org_state) references countries(state_tag));

create table products(
id_prod  number(3) primary key,
prod_name varchar(20) not null,
prod_type varchar(3),
sell_type varchar(3),
price number(4,1),
constraint type_fk foreign key(prod_type) references product_types(prod_type_tag));

create table receipts(
receipt_id number(8) primary key,
transaction_date date,
id_market number(4),
constraint mark_fk foreign key(id_market) references markets(id_market));

create table provide(
id_supplier number(3),
prod_type_tag varchar(3),
constraint provide_pk primary key (prod_type_tag,id_supplier),
constraint provide_id_fk foreign key (id_supplier) references suppliers(id_supplier),
constraint provide_tg_fk foreign key (prod_type_tag) references product_types(prod_type_tag));
```

```
create table contain(
receipt_id number(8),
id_prod number(3),
quantity number(4,2),
constraint contain_pk primary key (id_prod,receipt_id),
constraint contain_idp_fk foreign key (id_prod) references products(id_prod),
constraint contain_rid_fk foreign key (receipt_id) references receipts(receipt_id));

create table supply(
id_supplier number(3),
id_prod number(3),
id_market number (4),
constraint supply_pk primary key (id_supplier,id_prod,id_market),
constraint supply_ids_fk foreign key (id_supplier) references suppliers(id_supplier),
constraint supply_idp_fk foreign key (id_prod) references products(id_prod),
constraint supply_idm_fk foreign key (id_market) references markets(id_market));

create table know(
id_emp number(3),
lang_tag varchar(3),
constraint know_pk primary key (id_emp,lang_tag),
constraint know_ids_fk foreign key (id_emp) references employees(id_emp),
constraint know_idm_fk foreign key (lang_tag) references languages(lang_tag));
```

5. Popularea bazei de date

```
insert into countries
   values('ROM','Romania');
insert into countries
```

```
    values('TUR','Turkey');
insert into countries
    values('BLG','Bulgaria');
insert into countries
    values('HUN','Hungary');
insert into countries
    values('SPN','Spain');
insert into countries
    values('FRA','France');
insert into countries
    values('USA','United States of America');
insert into countries
    values('CHN','China');
insert into countries
    values('GER','Germany');
insert into countries
    values('LIB','Libya');
insert into countries
    values('LEB','Lebanon');
insert into countries
    values('RCO','Congo');
insert into countries
    values('DRC','Congo');
```

Script Output × Query Result ×

SQL | All Rows Fetched: 13 in 0.003 seconds

	STATE_TAG	STATE_NAME
1	ROM	Romania
2	TUR	Turkey
3	BLG	Bulgaria
4	HUN	Hungary
5	SPN	Spain
6	FRA	France
7	USA	United States of America
8	CHN	China
9	GER	Germany
10	LIB	Libya
11	LEB	Lebanon
12	RCO	Congo
13	DRC	Congo

```
insert into cities
   values ('BUC','Bucharest','ROM');
insert into cities
   values ('CTN','Constanta','ROM');
insert into cities
   values ('SOF','Sofia','BLG');
insert into cities
   values ('BDP','Budapest','HUN');
insert into cities
   values ('SZG','Szeged','HUN');
insert into cities
   values ('MAD','Madrid','SPN');
insert into cities
   values ('BAR','Barcelona','SPN');
insert into cities
   values ('VAL','Valencia','SPN');
insert into cities
```

```
   values ('PAR','Paris','FRA');
insert into cities
   values ('MRS','Marseilles','FRA');
insert into cities
   values ('LYN','Lyon','FRA');
insert into cities
   values ('BRL','Berlin','GER');
insert into cities
   values ('FRK','Frankfurt','GER');
insert into cities
   values ('MUN','Munich','GER');
insert into cities
   values ('HAM','Hamburg','GER');
insert into cities
   values ('TRP','Tripoli','LIB');
insert into cities
   values ('TRI','Tripoli','LEB');
```

Script Output x | Query Result x

SQL | All Rows Fetched: 17 in 0.003 seconds

	CITY...	CITY_NAME	STATE_TAG
1	BUC	Bucharest	ROM
2	CTN	Constanta	ROM
3	SOF	Sofia	BLG
4	BDP	Budapest	HUN
5	SZG	Szeged	HUN
6	MAD	Madrid	SPN
7	BAR	Barcelona	SPN
8	VAL	Valencia	SPN
9	PAR	Paris	FRA
10	MRS	Marseilles	FRA
11	LYN	Lyon	FRA
12	BRL	Berlin	GER
13	FRK	Frankfurt	GER
14	MUN	Munich	GER
15	HAM	Hamburg	GER
16	TRP	Tripoli	LIB
17	TRI	Tripoli	LEB

```
insert into addresses
   values(addresses_seq.nextval,'BUC','030353','Iuliu Maniu',5);
insert into addresses
   values(addresses_seq.nextval,'CTN','139303','Mihai Eminescu',23);
insert into addresses
   values(addresses_seq.nextval,'SOF','45-37-56','Aleksandry Zavdevsky',8);
insert into addresses
   values(addresses_seq.nextval,'BDP','678-205','Erkel',128);
insert into addresses
   values(addresses_seq.nextval,'SZG','723-365','Hatvan',52);
insert into addresses
   values(addresses_seq.nextval,'MAD','ZA56B3','Juan Bravo',12);
insert into addresses
   values(addresses_seq.nextval,'MAD','ZA67C9','Pedro de Valdivia',47);
```

```
insert into addresses
   values(addresses_seq.nextval,'VAL','GT49R2','Carrer de Borriana',5);
insert into addresses
   values(addresses_seq.nextval,'BAR','RH79N2','Avinguda Diagonal',34);
insert into addresses
   values(addresses_seq.nextval,'PAR','568230','Louis Rolland',5);
insert into addresses
   values(addresses_seq.nextval,'MRS','137475','Vincent Scotto',8);
insert into addresses
   values(addresses_seq.nextval,'LYN','567459','de Bonnel',76);
insert into addresses
   values(addresses_seq.nextval,'BRL','651246','Hallesches',89);
insert into addresses
   values(addresses_seq.nextval,'HAM','678578','Michelsenweg',35);
insert into addresses
   values(addresses_seq.nextval,'MUN','654345','Rosenheimer',87);
insert into addresses
   values(addresses_seq.nextval,'FRK','975773','Engelthaler',90);
insert into addresses
   values(addresses_seq.nextval,'BUD','498-275','Szytemlen',83);
```

Script Output | Query Result

SQL | All Rows Fetched: 16 in 0.003 seconds

	ADDRESS_CODE	CITY_TAG	POSTAL_CODE	STREET_NAME	NUMBER_S
1	20	BUC	030353	Iuliu Maniu	5
2	40	CTN	139303	Mihai Eminescu	23
3	60	SOF	45-37-56	Aleksandry Zavdevsky	8
4	80	BDP	678-205	Erkel	128
5	100	SZG	723-365	Hatvan	52
6	120	MAD	ZA56B3	Juan Bravo	12
7	140	MAD	ZA67C9	Pedro de Valdivia	47
8	160	VAL	GT49R2	Carrer de Borriana	5
9	180	BAR	RH79N2	Avinguda Diagonal	34
10	200	PAR	568230	Louis Rolland	5
11	220	MRS	137475	Vincent Scotto	8
12	240	LYN	567459	de Bonnel	76
13	260	BRL	651246	Hallesches	89
14	280	HAM	678578	Michelsenweg	35
15	300	MUN	654345	Rosenheimer	87
16	320	FRK	975773	Engelthaler	90

```
insert into departments
   values(departments_seq.nextval,'Bakery',2200);
insert into departments
   values(departments_seq.nextval,'Personnel',3500);
insert into departments
   values(departments_seq.nextval,'Sales',2300);
insert into departments
   values(departments_seq.nextval,'Marketing',3700);
insert into departments
   values(departments_seq.nextval,'Customer Service',3000);
insert into departments
   values(departments_seq.nextval,'Grocery',2200);
insert into departments
   values(departments_seq.nextval,'Logistics',3200);
insert into departments
   values(departments_seq.nextval,'Butcher',2200);
```

```
insert into departments
   values(departments_seq.nextval,'Board',5000);
```

Script Output x | Query Result x

SQL | All Rows Fetched: 9 in 0.003 seconds

	ID_DEPART	DEPARTMENTG_NAME	BASE_SALARY
1	20	Bakery	2200
2	30	Personnel	3500
3	40	Sales	2300
4	50	Marketing	3700
5	60	Customer Service	3000
6	70	Grocery	2200
7	80	Logistics	3200
8	90	Butcher	2200
9	100	Board	5000

```
insert into maintainance_companies
   values(maintainance_seq.nextval,'Luna si Bec','24-Apr-2018',2500,'Cleaning',null,14);
insert into maintainance_companies
   values(maintainance_seq.nextval,'Boris Clean','19-Feb-2019',3000,'Cleaning',null,7);
insert into maintainance_companies
   values(maintainance_seq.nextval,'ROSAFE','02-Sep-2018',3500,'Security','theft',null);
insert into maintainance_companies
   values(maintainance_seq.nextval,'Sparkling','04-May-2020',2700,'Cleaning',null,14);
insert into maintainance_companies
   values(maintainance_seq.nextval,'Feliz Limpio','23-Mar-2021',3200,'Cleaning',null,7);
insert into maintainance_companies
   values(maintainance_seq.nextval,'La Mapadora', '30-Aug-2019',3000,'Cleaning',null,7);
insert into maintainance_companies
   values(maintainance_seq.nextval,'Belle', '27-Oct-2020',2800,'Cleaning',null,14);
insert into maintainance_companies
   values(maintainance_seq.nextval,'Hermannstein', '10-Jun-2021',3500,'Cleaning',null,7);
insert into maintainance_companies
```

```
   values(maintainance_seq.nextval,'BGS Bulgaria', '13-Sep-2021',4000,'Security','theft',null);
insert into maintainance_companies
   values(maintainance_seq.nextval,'BGS Hungary', '28-Oct-2020',4200,'Security','safety',null);
insert into maintainance_companies
   values(maintainance_seq.nextval,'Garda Grande', '28-Mar-2019',3700,'Security','safety',null);
insert into maintainance_companies
   values(maintainance_seq.nextval,'Protection Gran', '17-Jul-2018',4300,'Security','safety',null);
insert into maintainance_companies
   values(maintainance_seq.nextval,'BGS Germany', '24-Nov-2020',4500,'Security','safety',null);
```

Script Output | Query Result

SQL | All Rows Fetched: 13 in 0.002 seconds

	ID_COMP	COMPANY_NAME	COLLAB_START_DATE	MONTHLY_PAYMENT	COMPANY_TYPE	MAIN_SPECIALISATION	CLEANING_FREQUENCY
1	10	Luna si Bec	24-APR-18	2500	Cleaning	(null)	14
2	12	Boris Clean	19-FEB-19	3000	Cleaning	(null)	7
3	14	ROSAFE	02-SEP-18	3500	Security	theft	(null)
4	16	Sparkling	04-MAY-20	2700	Cleaning	(null)	14
5	18	Feliz Limpio	23-MAR-21	3200	Cleaning	(null)	7
6	20	La Mapadora	30-AUG-19	3000	Cleaning	(null)	7
7	22	Belle	27-OCT-20	2800	Cleaning	(null)	14
8	24	Hermannstein	10-JUN-21	3500	Cleaning	(null)	7
9	26	BGS Bulgaria	13-SEP-21	4000	Security	theft	(null)
10	28	BGS Hungary	28-OCT-20	4200	Security	safety	(null)
11	30	Garda Grande	28-MAR-19	3700	Security	safety	(null)
12	32	Protection Gran	17-JUL-18	4300	Security	safety	(null)
13	34	BGS Germany	24-NOV-20	4500	Security	safety	(null)

```
insert into markets
   values(5,20,200,14,10);
insert into markets
   values(markets_seq.nextval,20,200,14,10);
insert into markets
   values(markets_seq.nextval,40,300,14,10);
insert into markets
   values(markets_seq.nextval,60,250,26,12);
insert into markets
   values(markets_seq.nextval,380,350,28,16);
insert into markets
```

```
   values(markets_seq.nextval,100,300,28,16);
insert into markets
   values(markets_seq.nextval,120,250,30,18);
insert into markets
   values(markets_seq.nextval,140,350,30,20);
insert into markets
   values(markets_seq.nextval,160,300,30,18);
insert into markets
   values(markets_seq.nextval,180,350,30,18);
insert into markets
   values(markets_seq.nextval,200,400,38,22);
insert into markets
   values(markets_seq.nextval,220,300,38,22);
insert into markets
   values(markets_seq.nextval,240,350,38,22);
insert into markets
   values(markets_seq.nextval,260,400,34,24);
insert into markets
   values(markets_seq.nextval,280,350,34,24);
insert into markets
   values(markets_seq.nextval,300,400,34,24);
insert into markets
   values(markets_seq.nextval,320,300,34,24);
```

Script Output | Query Result

SQL | All Rows Fetched: 13 in 0.002 seconds

	ID_MARKET	ADDRESS_CODE	SURFACE	ID_COMP_SEC	ID_COMP_CLE
1	10	20	200	14	10
2	15	40	300	14	10
3	20	60	250	26	12
4	30	100	300	28	16
5	35	120	250	30	18
6	40	140	350	30	20
7	45	160	300	30	18
8	50	180	350	30	18
9	70	260	400	34	24
10	75	280	350	34	24
11	80	300	400	34	24
12	85	320	300	34	24
13	5	20	200	14	10

```
insert into employees
   values(0,'Popescu','Mihnea','08-May-2019','07239890',null,40,5);
insert into employees
   values(emp_seq.nextval,'Popescu','Alexandru','02-May-2018','07235890',null,40,10);
insert into employees
   values(emp_seq.nextval,'Ionescu','Alina','03-May-2018','07635990',0.15,40,10);
insert into employees
   values(emp_seq.nextval,'Miron','Horatiu','07-May-2018','07635910',0.20,70,10);
insert into employees
   values(emp_seq.nextval,'Amirunei','Georgeta','05-May-2018','07655990',0.30,100,10);


insert into employees
   values(emp_seq.nextval,'Mihailescu','Adina','07-Jun-2019','07235890',0.25,60,15);
insert into employees
   values(emp_seq.nextval,'Ionescu','Alina','03-May-2018','07633990',null,40,15);
insert into employees
   values(emp_seq.nextval,'Miron','Horatiu','07-May-2018','07635990',0.05,90,15);
```

```
insert into employees
   values(emp_seq.nextval,'Amirunei','Georgeta','05-May-2018','07235990',0.15,100,15);

insert into employees
   values(emp_seq.nextval,'Milovich','Dmitry','02-May-2020','08235890',null,40,20);
insert into employees
   values(emp_seq.nextval,'Sayushkaya','Marya','03-Jul-2021','08635990',0.15,70,20);
insert into employees
   values(emp_seq.nextval,'Dobrovich','Aleksandr','07-Aug-2020','08635990',0.20,100,20);
insert into employees
   values(emp_seq.nextval,'Milovich','Dmitry','02-May-2020','08235890',null,220,20);
insert into employees
   values(emp_seq.nextval,'Sayushkaya','Marya','03-Jul-2021','08635990',0.15,240,20);
insert into employees
   values(emp_seq.nextval,'Dobrovich','Aleksandr','07-Aug-2020','08635990',0.20,100,20);

insert into employees
   values(emp_seq.nextval,'Anglossy','Istvan','05-Apr-2021','09235890',null,40,30);
insert into employees
   values(emp_seq.nextval,'Janos','Ardony','08-Jul-2021','09235890',null,40,30);
insert into employees
   values(emp_seq.nextval,'Gyorgethery','Marika','05-Apr-2021','09235890',null,100,30);

insert into employees
   values(emp_seq.nextval,'Garcia','Esteban','06-Aug-2021','04235890',null,40,35);
insert into employees
   values(emp_seq.nextval,'LLuro','Javier','07-Sep-2020','04235070',null,100,35);

insert into employees
```

```
   values(emp_seq.nextval,'Fernan','Rofrigo','15-Apr-2021','04215899',null,100,40);
insert into employees
   values(emp_seq.nextval,'Goices','Ramiriz','15-Jul-2021','04215899',null,40,40);

insert into employees
   values(emp_seq.nextval,'Fernan','Jimena','15-Apr-2021','04215792',null,50,45);
insert into employees
   values(emp_seq.nextval,'Garcia','Lopez','17-Apr-2021','04215891',null,40,45);
insert into employees
   values(emp_seq.nextval,'Malfrida','Infanta','18-Apr-2021','04215991',0.15,100,45);

insert into employees
   values(emp_seq.nextval,'Alvarez','Rofrigo','15-Apr-2021','04215896',null,40,50);
insert into employees
   values(emp_seq.nextval,'Garcia','Lopez','17-Apr-2021','04215871',null,100,50);

insert into employees
   values(emp_seq.nextval,'Boivelle','Marie','15-Apr-2021','04215895',null,100,55);
insert into employees
   values(emp_seq.nextval,'Blois','isabelle','17-Apr-2021','04285891',null,40,55);

insert into employees
   values(emp_seq.nextval,'Marquie','Alessandre','15-Apr-2021','04215893',null,40,60);
insert into employees
   values(emp_seq.nextval,'Broget','Almec','17-Apr-2021','04225892',null,100,60);

insert into employees
   values(emp_seq.nextval,'Charlee','Antoine','15-Apr-2021','04215866',null,40,65);
insert into employees
```

values(emp_seq.nextval,'Vivizon','Louis','17-Apr-2021','04215867',null,100,65);

insert into employees

values(emp_seq.nextval,'Gustav','Klauss','17-Apr-2021','09215867',null,40,70);

insert into employees

values(emp_seq.nextval,'Mahstern','Mark','17-Apr-2021','09315867',null,40,75);

insert into employees

values(emp_seq.nextval,'Friedrichson','Karl','17-Apr-2021','09415867',null,40,80);

Script Output x | Query Result x

SQL | All Rows Fetched: 25 in 0.004 seconds

	ID_EMP	LAST_NAME	FIRST_NAME	HIRE_DATE	PHONE	COMMISSION_QUOEF	ID_DEPART	ID_MARKET
1	1	Popescu	Alexandru	02-MAY-18	07235890	(null)	40	10
2	2	Ionescu	Alina	03-MAY-18	07635990	0.15	40	10
3	3	Miron	Horatiu	07-MAY-18	07635910	0.2	70	10
4	4	Amirunei	Georgeta	05-MAY-18	07655990	0.3	100	10
5	6	Ionescu	Alina	03-MAY-18	07633990	(null)	40	15
6	8	Amirunei	Georgeta	05-MAY-18	07235990	0.15	100	15
7	9	Milovich	Dmitry	02-MAY-20	08235890	(null)	40	20
8	10	Sayushkaya	Marya	03-JUL-21	08635990	0.15	70	20
9	15	Anglossy	Istvan	05-APR-21	09235890	(null)	40	30
10	18	Garcia	Esteban	06-AUG-21	04235890	(null)	40	35
11	19	LLuro	Javier	07-SEP-20	04235070	(null)	100	35
12	20	Fernan	Rofrigo	15-APR-21	04215899	(null)	100	40
13	22	Fernan	Jimena	15-APR-21	04215792	(null)	50	45
14	23	Garcia	Lopez	17-APR-21	04215891	(null)	40	45
15	24	Malfrida	Infanta	18-APR-21	04215991	0.15	100	45
16	25	Alvarez	Rofrigo	15-APR-21	04215896	(null)	40	50
17	26	Garcia	Lopez	17-APR-21	04215871	(null)	100	50
18	33	Gustav	Klauss	17-APR-21	09215867	(null)	40	70

insert into languages

values('ROM','Romanian');

insert into languages

values('BLG','Bulgarian');

insert into languages

```
  values('FRE','French');
insert into languages
  values('ENG','English');
insert into languages
  values('DEU','German');
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.002 seconds

	LANG_TAG	LANG_NAME
1	ROM	Romanian
2	BLG	Bulgarian
3	FRE	French
4	ENG	English
5	DEU	German

```
insert into product_types
  values('MEA','Meats');
insert into product_types
  values('DRY','Dairy');
insert into product_types
  values('FRU','Fruits');
insert into product_types
  values('VEG','Vegetables');
insert into product_types
  values('CSM','Cosmetics');
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.003 seconds

	PROD_TYPE_TAG	PROD_NAME
1	MEA	Meats
2	DRY	Dairy
3	FRU	Fruits
4	VEG	Vegetables
5	CSM	Cosmetics

```
insert into suppliers
```

```
   values(100,'Ferma lui Ion','ROM');
insert into suppliers
   values(101,'Antalya Garden','TUR');
insert into suppliers
   values(102,'Wuhan Industrials','CHN');
insert into suppliers
   values(103,'Agricola Sevilla','SPN');
insert into suppliers
   values(104,'Elmiplant','FRA');
insert into suppliers
   values(105,'Kasimiva Fermata','BLG');
insert into suppliers
   values(106,'Elmiplant','SPN');
```

Script Output x | Query Result x

SQL | All Rows Fetched: 7 in 0.003 seconds

	ID_SUPPLIER	NAME_S	ORG_STATE
1	100	Ferma lui Ion	ROM
2	101	Antalya Garden	TUR
3	102	Wuhan Industrials	CHN
4	103	Agricola Sevilla	SPN
5	104	Elmiplant	FRA
6	105	Kasimiva Fermata	BLG
7	106	Elmiplant	SPN

```
insert into products
   values(100,'carrots','VEG','eng',1.2);
insert into products
   values(101,'carrots','VEG','eng',1.6);
insert into products
   values(102,'soap','CSM','end',5);
insert into products
```

```
   values(103,'shampoo','CSM','end',10);
insert into products
   values(104,'beef','MEA','end',8);
insert into products
   values(105,'fried chicken','MEA','end',10);
insert into products
   values(106,'apples','FRU','eng',2.8);
insert into products
   values(107,'cheese','DRY','end',5);
insert into products
   values(108,'milk','DRY','end',6);
insert into products
   values(109,'face cream','CSM','end',12);
insert into products
   values(110,'hand cream','CSM','end',12);
insert into products
   values(111,'pears casolette','FRU','end',3);
insert into products
   values(112,'potatoes sack','VEG','end',4);
insert into products
   values(113,'hand cream','CSM','end',10);
insert into products
   values(114,'bananas','FRU','eng',2.5);
```

Script Output x | Query Result x

SQL | All Rows Fetched: 15 in 0.004 seconds

	ID_PROD	PROD_NAME	PROD_TYPE	SELL_TYPE	PRICE
1	100	carrots	VEG	eng	1.2
2	101	carrots	VEG	eng	1.6
3	102	soap	CSM	end	5
4	103	shampoo	CSM	end	10
5	104	beef	MEA	end	8
6	105	fried chicken	MEA	end	10
7	106	apples	FRU	eng	2.8
8	107	cheese	DRY	end	5
9	108	milk	DRY	end	6
10	109	face cream	CSM	end	12
11	110	hand cream	CSM	end	13.2
12	111	pears casolette	FRU	end	3
13	112	potatoes sack	VEG	end	4
14	113	hand cream	CSM	end	11
15	114	bananas	FRU	eng	2.5

```
insert into receipts
   values(10,'04-May-2022',35);
insert into receipts
   values(11,'04-May-2022',35);
insert into receipts
   values(12,'04-May-2022',35);
insert into receipts
   values(13,'04-May-2022',35);
insert into receipts
   values(14,'04-May-2022',35);
insert into receipts
   values(15,'04-May-2022',35);
insert into receipts
   values(16,'05-May-2022',35);
insert into receipts
```

```
   values(17,'06-May-2022',35);
insert into receipts
   values(18,'07-May-2022',35);
insert into receipts
   values(19,'07-May-2022',30);
```

Script Output x | Query Result x

SQL | All Rows Fetched: 10 in 0.003 seconds

	RECEIPT_ID	TRANSACTION_DATE	ID_MARKET
1	10	04-MAY-22	35
2	11	04-MAY-22	35
3	12	04-MAY-22	35
4	13	04-MAY-22	35
5	14	04-MAY-22	35
6	15	04-MAY-22	35
7	16	05-MAY-22	35
8	17	06-MAY-22	35
9	18	07-MAY-22	35
10	19	07-MAY-22	30

```
insert into provide
   values(100,'MEA');
insert into provide
   values(100,'DRY');
insert into provide
   values(102,'CSM');
insert into provide
   values(105,'VEG');
insert into provide
```

```
   values(105,'FRU');
insert into provide
   values(104,'CSM');
insert into provide
   values(103,'VEG');
insert into provide
   values(103,'FRU');
insert into provide
   values(103,'DRY');
insert into provide
   values(101,'VEG');
insert into provide
   values(101,'FRU');
insert into provide
   values(101,'MEA');
```

Script Output x | Query Result x

SQL | All Rows Fetched: 12 in 0.003 seconds

	ID_SUPPLIER	PROD_TYPE_TAG
1	102	CSM
2	104	CSM
3	100	DRY
4	103	DRY
5	101	FRU
6	103	FRU
7	105	FRU
8	100	MEA
9	101	MEA
10	101	VEG
11	103	VEG
12	105	VEG

```
insert into contain
   values(10,100,5);
insert into contain
```

```
    values(10,101,5);
insert into contain
    values(11,103,2);
insert into contain
    values(12,111,3);
insert into contain
    values(13,109,1);
insert into contain
    values(13,110,1);
insert into contain
    values(14,104,2);
insert into contain
    values(15,106,3);
insert into contain
    values(16,108,8);
insert into contain
    values(16,105,1);
insert into contain
    values(17,114,2);
```

Script Output x | Query Result x

SQL | All Rows Fetched: 11 in 0.003 seconds

	RECEIPT_ID	ID_PROD	QUANTITY
1	10	100	5
2	10	101	5
3	11	103	2
4	12	111	3
5	13	109	1
6	13	110	1
7	14	104	2
8	15	106	3
9	16	108	8
10	16	105	1
11	17	114	2

```
insert into supply
   values(100,104,10);
insert into supply
   values(100,105,10);
insert into supply
   values(100,104,15);
insert into supply
   values(100,107,10);
insert into supply
   values(100,108,10);
insert into supply
   values(100,108,15);
insert into supply
   values(102,109,10);
insert into supply
   values(102,110,10);
insert into supply
   values(102,110,35);
```

```
insert into supply
   values(105,112,10);
insert into supply
   values(105,106,10);
insert into supply
   values(105,106,15);
```

Script Output | Query Result

SQL | All Rows Fetched: 12 in 0.003 seconds

	ID_SUPPLIER	ID_PROD	ID_MARKET
1	100	104	10
2	100	104	15
3	100	105	10
4	100	107	10
5	100	108	10
6	100	108	15
7	102	109	10
8	102	110	10
9	102	110	35
10	105	106	10
11	105	106	15
12	105	112	10

```
insert into know
   values(10,'BLG');
insert into know
   values(10,'ENG');
insert into know
   values(10,'FRE');
insert into know
   values(2,'BLG');
insert into know
   values(2,'ENG');
insert into know
   values(2,'FRE');
```

```
insert into know
   values(20,'BLG');
insert into know
   values(20,'ENG');
insert into know
   values(20,'FRE');
insert into know
   values(8,'BLG');
insert into know
   values(8,'ENG');
insert into know
   values(9,'FRE');
```

	ID_EMP	LANG_TAG
1	2	BLG
2	2	ENG
3	2	FRE
4	8	BLG
5	8	ENG
6	9	FRE
7	10	BLG
8	10	ENG
9	10	FRE
10	20	BLG
11	20	ENG
12	20	FRE

6. **Formulați în limbaj natural o problemă pe care să orezolvați folosind unsubprogram stocatindependent care să utilizeze douătipuridiferite de colecțiistudiate. Apelați subprogramul.**

Cerinta: Pentru un oras specificat sa se afiseze numarul de angajati care lucreaza acolo, precum si date despre acestia: nume, prenume, salariu de baza si numele departamentului la care lucreaza

--6 colectiile varray si tablou indexat + tip de date compus(record),

```
--join-uri si tratarea exceptiilor
create or replace procedure display_emplpoyees_city(oras cities.city_name%type) is
   type emp_rec is record( prenume employees.first_name%type,
                           nume employees.last_name%type,
                           department departments.departmentg_name%type,
                           salariu departments.base_salary%type);
   type tablou_indexat is table of emp_rec index by pls_integer;
   type vector_id_emp is varray(100) of employees.id_emp%type;
   v_cod_oras cities.city_name%type;
   v_ids vector_id_emp;
   tab_emp tablou_indexat;
begin
   select city_tag into v_cod_oras from cities where upper(oras) = upper(city_name);

   select employees.id_emp bulk collect into v_ids
   from cities join (addresses) using (city_tag)
            join markets using(address_code)
               join employees using (id_market)
   where city_tag = v_cod_oras;

   for i in v_ids.first..v_ids.last loop
      select first_name, last_name, departmentg_name, base_salary
      into tab_emp(i)
      from employees join departments using (id_depart)
      where id_emp = v_ids(i);
   end loop;

   DBMS_OUTPUT.PUT_LINE('In orasul ' || oras || ' lucreaza ' || tab_emp.count || ' angajati');
```

```
    for i in tab_emp.first..tab_emp.last loop
        DBMS_OUTPUT.PUT_LINE(tab_emp(i).prenume || ' ' || tab_emp(i).nume || ', cu salariul de baza ' || tab_emp(i).salariu || ' la departamentul ' || tab_emp(i).department);
    end loop;

exception
    when TOO_MANY_ROWS then DBMS_OUTPUT.PUT_LINE('Mai multe orase cu numele ' || oras || '!');
    when NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Orasul ' || oras || ' nu exista!');
    when OTHERS then DBMS_OUTPUT.PUT_LINE('Eroare necunoscuta!');
end display_emplpoyees_city;
/
begin
    display_emplpoyees_city('Tripoli');
end;
/
```

```
Mai multe orase cu numele Tripoli!


PL/SQL procedure successfully completed.
```

```
begin
    display_emplpoyees_city('New York');
end;
/
```

```
Orasul New York nu exista!


PL/SQL procedure successfully completed.
```

```
begin
```

```
   display_emplpoyees_city('Constanta');
end;
/
```

```
In orasul Constanta lucreaza 2 angajati
Georgeta Amirunei, cu salariul de baza 5000 la departamentul Board
Alina Ionescu, cu salariul de baza 2300 la departamentul Sales


PL/SQL procedure successfully completed.
```

7. Formulați în limbaj natural o problemă pe care să orezolvați folosind un subprogram stocatindependent care să utilizeze 2tipuridiferite de cursoarestudiate, unul dintre acestea fiind cursor parametrizat. Apelați subprogramul.

Cerinta: Date fiind un nume de oras si un numar natural, obtineti magazinele din tara in care se afla orasul respectiv care au suprafata mai mare sau egala cu numarul natural dat. Afisati pentru fiecare magazin id-ul si numarul de angajati.

```
--7 ciclu cursor parametrizat + cursor clasic
create or replace procedure nr_ang_extrem_stat (nume_oras cities.city_name%type, selectie number) is
   subquery exception;
   pragma EXCEPTION_INIT(subquery, -01427);
   cursor c_magazine (nume_tara countries.state_name%type) is
      select id_market, count (*) nr_ang
      from (select id_market, id_emp
            from countries co
               join cities ci on (ci.state_tag = co.state_tag and upper(co.state_name) = upper(nume_tara))
               join addresses using (city_tag)
               join markets using (address_code)
               join employees using(id_market))
      group by id_market;
   cursor c_suprafata is
```

```
      select *
      from markets
      where surface >= selectie;
   type tab_indexat is table of markets%rowtype index by pls_integer;
   v_state_name countries.state_name%type;
   v_id_market markets.id_market%type;
   v_surface markets.surface%type;
   v_market markets%rowtype;
   v_index number := 1;
   v_tab tab_indexat;
begin
   select state_name into v_state_name from countries
   where (select state_tag
         from cities
         where upper(city_name) = upper(nume_oras)) = state_tag;

   if selectie > 0 then
      open c_suprafata;

      loop
         fetch c_suprafata into v_tab(v_index);
         exit when c_suprafata%NOTFOUND;
         v_index := v_index + 1;
      end loop;

      close c_suprafata;

      for rec in c_magazine(v_state_name) loop
         for i in v_tab.first..v_tab.last loop
```

```
            if v_tab(i).id_market = rec.id_market
              then DBMS_OUTPUT.PUT_LINE('Magazinul cu id-ul ' || rec.id_market || ' si ' || rec.nr_ang || ' angajati are suprafata de ' || v_tab(i).surface || ' mp');
            end if;
          end loop;
        end loop;
    else
        DBMS_OUTPUT.PUT_LINE('Optiune incorecta pentru selectia suprafetei');
    end if;
exception
    when NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Orasul nu exista!');
    when subquery then DBMS_OUTPUT.PUT_LINE('Mai multe orase cu acest nume!');
    when OTHERS then DBMS_OUTPUT.PUT_LINE('Eroare necunoscuta!');
end nr_ang_extrem_stat;
/

begin
    nr_ang_extrem_stat('Bucharest', 100);
end;
/
```

```
Magazinul cu id-ul 5 si 1 angajati are suprafata de 200 mp
Magazinul cu id-ul 10 si 8 angajati are suprafata de 200 mp
Magazinul cu id-ul 15 si 2 angajati are suprafata de 300 mp


PL/SQL procedure successfully completed.
```

```
begin
    nr_ang_extrem_stat('Bucharest', -1);
end;
/
```

```
Optiune incorecta pentru selectia suprafetei


PL/SQL procedure successfully completed.
```

```
begin
   nr_ang_extrem_stat('New York', 150);
end;
/
```

```
Orasul nu exista!


PL/SQL procedure successfully completed.
```

```
begin
   nr_ang_extrem_stat('Tripoli', 200);
end;
/
```

```
Mai multe orase cu acest nume!


PL/SQL procedure successfully completed.
```

8. Formulați în limbaj natural o problemă pe care să orezolvați folosind un subprogram stocatindependent de tip funcțiecare să utilizeze într-o singură comandă SQL3 dintre tabelele definite. Definiți minim 2 excepții. Apelați subprogramulastfel încât să evidențiați toatecazurile tratate.

Cerinta: Stabiliti pentru un magazin al carui cod este dat daca acesta este local sau international. Magazinele locale au minim jumatate + 1 din totalul produselor comercializate de origine locala, adica furnizate de producatori din acelasi stat in care se afla si magazinul, altfel sunt internationale. Implementati o functie pentru rezolvarea problemei care sa returneze o valoare de adevar.

```
--8 functie cu cerere in SQL care foloseste 3 tabele (in cazul de fata
--operatia de join pe 6 tabele), returning boolean, tablouri imbricate si indexate,
```

```
--outer join, functia nvl, tip de date compus(record), bulk collect into
create or replace function este_local (cod markets.id_market%type) return boolean is
   fara_return exception;
   cursor c_nr_prod is
   select id_market, nvl(nr_prod,0) nr_prod
   from (select id_market, count(*) nr_prod
        from supply
        group by id_market) full outer join markets using(id_market)
   order by id_market;
   type prod_rec is record(id markets.id_market%type,
                           nr_prod number);
   type tablou_indexat is table of prod_rec index by pls_integer;
   type tablou_imbricat is table of prod_rec;
   v_locale tablou_indexat;
   v_totale tablou_imbricat := tablou_imbricat();
begin
   select id_market, nvl(nr_prod,0)
   bulk collect into v_locale
   from   (select id_market, count(*) nr_prod
           from supply
           join suppliers using (id_supplier)
           join markets using (id_market)
           join addresses using (address_code)
           join cities using (city_tag)
           join countries using (state_tag)
           where org_state = state_tag
           group by id_market)
           full outer join markets using(id_market)
   order by id_market;
```

```
    open c_nr_prod;

    loop
        v_totale.extend;
        fetch c_nr_prod into v_totale(v_totale.last);
        exit when c_nr_prod%NOTFOUND;
    end loop;

    close c_nr_prod;

    for i in v_locale.first..v_locale.last loop
        if (v_locale(i).id = cod) then
            if (v_locale(i).nr_prod / v_totale(i).nr_prod > 0.5) then
                return true;
            else
                if (v_locale(i).nr_prod / v_totale(i).nr_prod <= 0.5) then
                    return false;
                end if;
            end if;
        end if;
    end loop;

    raise fara_return;

exception
    --when NO_DATA_FOUND
    --    then DBMS_OUTPUT.PUT_LINE('Introduceti un cod valid!');
    --    return null;
```

```
    when ZERO_DIVIDE
        then DBMS_OUTPUT.PUT_LINE('Magazinul inca nu este aprovizionat!');
        return null;
    when fara_return
        then DBMS_OUTPUT.PUT_LINE('Introduceti un cod valid!');
        return null;
    when OTHERS
        then DBMS_OUTPUT.PUT_LINE('Eroare necunoscuta!');
        return null;
    --cazul TOO_MANY_ROWS nu are sens deoarece parametrul este un id
    --cazul NO_DATA_FOUND nu are sens deoarece, din structura functiei,
    --un cod care nu e valid genereaza un cursor gol deci se ajunge la final
    --fara sa se returneze ceva, nu se intra in cele doua foruri
end este_local;
/

declare
    v_status boolean;
begin
    v_status := este_local(15); --local

    if (v_status = true) then
        DBMS_OUTPUT.PUT_LINE('Magazin local');
    else
        DBMS_OUTPUT.PUT_LINE('Magazin international');
    end if;
end;
/
```

```
Magazin local

PL/SQL procedure successfully completed.
```

```
declare
   v_status boolean;
begin
   v_status := este_local(10); --international

   if (v_status = true) then
      DBMS_OUTPUT.PUT_LINE('Magazin local');
   end if;
   if (v_status = false) then
      DBMS_OUTPUT.PUT_LINE('Magazin international');
   end if;
end;
/
```

```
Magazin international

PL/SQL procedure successfully completed.
```

```
declare
   v_status boolean;
begin
   v_status := este_local(20); --fara produse

   if (v_status = true) then
      DBMS_OUTPUT.PUT_LINE('Magazin local');
   end if;
   if (v_status = false) then
```

```
      DBMS_OUTPUT.PUT_LINE('Magazin international');
   end if;
end;
/
```

```
Magazinul inca nu este aprovizionat!


PL/SQL procedure successfully completed.
```

```
declare
   v_status boolean;
begin
   v_status := este_local(17); --codul nu exista


   if (v_status = true) then
      DBMS_OUTPUT.PUT_LINE('Magazin local');
   end if;
   if (v_status = false) then
      DBMS_OUTPUT.PUT_LINE('Magazin international');
   end if;
end;
/
```

```
Introduceti un cod valid!


PL/SQL procedure successfully completed.
```

9. Formulați în limbaj natural o problemă pe care să orezolvați folosind un subprogram stocat independent de tip procedurăcare să utilizezeîntr-o singură comandăSQL5 dintre tabelele definite. Tratați toate excepțiile care pot apărea,

incluzând excepțiile NO_DATA_FOUND și TOO_MANY_ROWS. Apelați subprogramulastfel încât să evidențiați toate cazurile tratate.

Cerinta: Stabiliti daca produsele unui furnizor se comercializeaza intr-un stat dat. Implementati o procedura cu parametru de iesire

```
--9 procedura cu cerere in SQL care foloseste 3 tabele (in cazul de fata

--operatia de join pe 6 tabele), parametru de iesire, ciclu cursor cu subcereri

--definirea exceptiilor specifice pentru cazurile TOO_MANY_ROWS, NO_DATA_FOUND pentru fiecare parametru

create or replace procedure se_comercializeaza_in (furnizor in suppliers.name_s%type, tara in countries.state_name%type, status out boolean) is

   --definim exceptii noi ca sa tratam cazurile de exceptie pentru fiecare parametru

   no_furnizor exception;

   too_many_furnizor exception;

   no_stat exception;

   too_many_stat exception;

   v_exista number := 0;

   v_nr_rez number;

begin

   select count (*) into v_nr_rez

      from (select state_tag

           from countries

           where upper(state_name) = upper(tara));


   if (v_nr_rez > 1)

      then raise too_many_stat;

   end if;


   if (v_nr_rez < 1)

      then raise no_stat;
```

```
end if;

select count (*) into v_nr_rez
   from (select id_supplier
         from suppliers
         where upper(name_s) = upper(furnizor));

if (v_nr_rez > 1)
   then raise too_many_furnizor;
end if;

if (v_nr_rez < 1)
   then raise no_furnizor;
end if;

for i in   (select id_supplier--, count(id_supplier) nr_prod
           from supply
           join suppliers using (id_supplier)
           join markets using (id_market)
           join addresses using (address_code)
           join cities using (city_tag)
           join countries using (state_tag)
           where (upper(tara) = upper(state_name)
           and  upper(name_s) = upper(furnizor))
           --group by id_supplier
           order by state_tag) loop
   status := true;
   v_exista := v_exista + 1;
end loop;
```

```
    if v_exista = 0 then
        status := false;
        DBMS_OUTPUT.PUT_LINE('Furnizorul nu are produse pe piata!');
    else
        DBMS_OUTPUT.PUT_LINE('Furnizorul are produse pe piata!');
    end if;

exception
    when no_furnizor then
        DBMS_OUTPUT.PUT_LINE('Furnizor inexistent!');
        status := null;
    when too_many_furnizor then
        DBMS_OUTPUT.PUT_LINE('Mai multi furnizori cu acelasi nume!');
        status := null;
    when no_stat then
        DBMS_OUTPUT.PUT_LINE('Stat inexistent!');
        status := null;
    when too_many_stat then
        DBMS_OUTPUT.PUT_LINE('Mai multe state cu acelasi nume!');
        status := null;
    --when others then
    --    DBMS_OUTPUT.PUT_LINE('Eroare neidentificata!');
    --    status := null;
end se_comercializeaza_in;
/

declare
    este boolean;
```

```
begin
   se_comercializeaza_in('Elmiplant', 'Romania', este); --2 furnizori

   if este = true then
      DBMS_OUTPUT.PUT_LINE('Confirmare, se comercializeaza!');
   end if;

   if este = false then
      DBMS_OUTPUT.PUT_LINE('Confirmare, nu se comercializeaza!');
   end if;
end;
/
```

```
Mai multi furnizori cu acelasi nume!

PL/SQL procedure successfully completed.
```

```
declare
   este boolean;
begin
   se_comercializeaza_in('Ferma lui Ion', 'Congo', este); --2 state

   if este = true then
      DBMS_OUTPUT.PUT_LINE('Confirmare, se comercializeaza!');
   end if;

   if este = false then
      DBMS_OUTPUT.PUT_LINE('Confirmare, nu se comercializeaza!');
   end if;
end;
```

/

```
Mai multe state cu acelasi nume!


PL/SQL procedure successfully completed.
```

```
declare
   este boolean;
begin
   se_comercializeaza_in('Dero', 'Romania', este); --0 furnizori


   if este = true then
      DBMS_OUTPUT.PUT_LINE('Confirmare, se comercializeaza!');
   end if;


   if este = false then
      DBMS_OUTPUT.PUT_LINE('Confirmare, nu se comercializeaza!');
   end if;
end;
/
```

```
Furnizor inexistent!


PL/SQL procedure successfully completed.
```

```
declare
   este boolean;
begin
   se_comercializeaza_in('Ferma lui Ion', 'Japonia', este); --0 state


   if este = true then
      DBMS_OUTPUT.PUT_LINE('Confirmare, se comercializeaza!');
```

```
    end if;

    if este = false then
        DBMS_OUTPUT.PUT_LINE('Confirmare, nu se comercializeaza!');
    end if;
end;
/
```

```
Stat inexistent!


PL/SQL procedure successfully completed.
```

```
declare
    este boolean;
begin
    se_comercializeaza_in('Ferma lui Ion', 'Romania', este); -- da

    if este = true then
        DBMS_OUTPUT.PUT_LINE('Confirmare, se comercializeaza!');
    end if;

    if este = false then
        DBMS_OUTPUT.PUT_LINE('Confirmare, nu se comercializeaza!');
    end if;
end;
/
```

```
Furnizorul are produse pe piata!
Confirmare, se comercializeaza!


PL/SQL procedure successfully completed.
```

```
declare
   este boolean;
begin
   se_comercializeaza_in('Ferma lui Ion', 'Spain', este); -- nu


   if este = true then
      DBMS_OUTPUT.PUT_LINE('Confirmare, se comercializeaza!');
   end if;


   if este = false then
      DBMS_OUTPUT.PUT_LINE('Confirmare, nu se comercializeaza!');
   end if;
end;
/
```

```
Furnizorul nu are produse pe piata!
Confirmare, nu se comercializeaza!


PL/SQL procedure successfully completed.
```

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

Cerinta: Pentru a regla preturile dinamic, in functie de cerere, permiteti modificarea detaliilor produselor doar in timpul programului de lucru al magazinului, acesta fiind 10:00-21:00 de luni pana sambata si 11:00-19:00 duminica

```
--10 trigger LMD la nivel de comanda cu lansare de exceptii in script output
create or replace trigger casa_deschisa before delete or update on products
begin
   if to_char(sysdate, 'day') = 'SUNDAY' THEN
      if to_number(to_char(sysdate, 'hh24')) < 11 then
```

```
        RAISE_APPLICATION_ERROR(-20001, 'Magazinele se deschid dupa ora 11:00 duminica!');
      end if;

      if to_number(to_char(sysdate, 'hh24')) >= 19 then
        RAISE_APPLICATION_ERROR(-20002, 'magazinele se inchid dupa ora 19:00 in weekend!');
      end if;

    else
      if to_number(to_char(sysdate, 'hh24')) < 10 then
        RAISE_APPLICATION_ERROR(-20003, 'Magazinele se deschid dupa ora 10:00 de luni pana
sambata!');
      end if;

      if to_number(to_char(sysdate, 'hh24')) >= 21 then
        RAISE_APPLICATION_ERROR(-20004, 'Magazinele se inchid dupa ora 21:00 de luni pana sambata!');
      end if;

    end if;
end;
/

--alter trigger casa_deschisa disable;
--alter trigger casa_deschisa enable;

begin
    update products
    set price = price + price * 0.10
    where prod_name = 'hand cream';
end;
```

/

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 15 in 0.002 seconds

	ID_PROD	PROD_NAME	PROD_TYPE	SELL_TYPE	PRICE
1	100	carrots	VEG	eng	1.2
2	101	carrots	VEG	eng	1.6
3	102	soap	CSM	end	5
4	103	shampoo	CSM	end	10
5	104	beef	MEA	end	8
6	105	fried chicken	MEA	end	10
7	106	apples	FRU	eng	2.8
8	107	cheese	DRY	end	5
9	108	milk	DRY	end	6
10	109	face cream	CSM	end	12
11	110	hand cream	CSM	end	14.5
12	111	pears casolette	FRU	end	3
13	112	potatoes sack	VEG	end	4
14	113	hand cream	CSM	end	12.1
15	114	bananas	FRU	eng	2.5

```
begin
   delete from products
   where prod_name = 'beef';
end;
/
```

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

Cerinta: Orice magazin are nevoie de cel putin un vanzator(departamentul Sales) si un manager(departamentul Board) pentru a functiona. Creati un trigger care

nu permite adaugarea in baza de date a unui nou angajat la un magazin care nu are cele doua functii necesare daca anagajatul nu le poate indeplini

De exemplu, nu se permite angajarea unui macelar (departamentul Butcher) la un magazin X daca magazinul X nu are un manager si un vanzator.

```
--11 trigger LMD la nivel de linie cu lansare de exceptii in script output

create or replace trigger roluri_necesare_magazin before insert on employees for each row

declare

   v_nr_sales number;

   v_nr_board number;

   v_id_old_depart departments.id_depart%type;

   v_id_sales departments.id_depart%type;

   v_id_board departments.id_depart%type;

   sales_not_enough_insert exception;

   board_not_enough_insert exception;

begin

   select id_depart

   into v_id_sales

   from departments

   where upper(departmentg_name) = upper('Sales');


   select id_depart

   into v_id_board

   from departments

   where upper(departmentg_name) = upper('Board');


   select count(*)

   into v_nr_sales

   from markets

   join employees using(id_market)
```

```
   join departments using(id_depart)
   where id_market = :new.id_market
   and id_depart = v_id_sales;

   select count(*)
   into v_nr_board
   from markets
   join employees using(id_market)
   join departments using(id_depart)
   where id_market = :new.id_market
   and id_depart = v_id_board;

   if (v_nr_sales < 1 and :new.id_depart != v_id_sales) then
       RAISE_APPLICATION_ERROR(-20001, 'Angajati mai intai un vanzator!');
   end if;

   if (v_nr_board < 1 and :new.id_depart != v_id_board) then
      if (v_nr_sales > 0 and :new.id_depart = v_id_sales) then
          RAISE_APPLICATION_ERROR(-20002, 'Angajati mai intai un manager!');
      end if;
   end if;
end;
/
begin
   insert into employees   --mai intai manager
      values(4000,'Andrei','Mihailescu','08-May-2019','08239890',null,40,5);
end;
/
```

```
end;
Error report -
ORA-20002: Angajati mai intai un manager!
ORA-06512: at "UTILIZ.ROLURI_NECESARE_MAGAZIN", line 42
ORA-04088: error during execution of trigger 'UTILIZ.ROLURI_NECESARE_MAGAZIN'
ORA-06512: at line 2
```

```
begin
   insert into employees   --mai intai un vanzator
      values(emp_seq.nextval,'Klogge','Frau','17-Apr-2021','09515867',null,80,85);
end;
/
```

```
Error report -
ORA-20001: Angajati mai intai un vanzator!
ORA-06512: at "UTILIZ.ROLURI_NECESARE_MAGAZIN", line 37
ORA-04088: error during execution of trigger 'UTILIZ.ROLURI_NECESARE_MAGAZIN'
ORA-06512: at line 2
```

```
begin
   insert into employees   --inserare fara erori
      values(2000,'Andrei','Mihailescu','08-May-2019','01209800',null,70,10);
end;
/
```

```
548
549 select * from employees;
550
551 rollback;
552
553 --12 trigger LDD cu lansare de exceptii in script output cu salvare de informatii
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 26 in 0.003 seconds

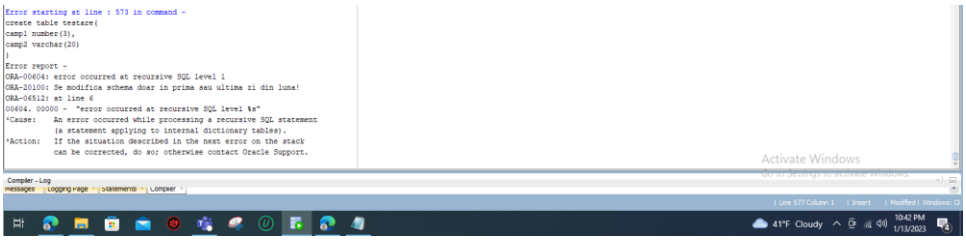
	ID_EMP	LAST_NAME	FIRST_NAME	HIRE_DATE	PHONE	COMMISSION_QUOEF	ID_DEPART	ID_MARKET
9	15	Anglossy	Istvan	05-APR-21	09235890	(null)	40	30
10	18	Garcia	Esteban	06-AUG-21	04235890	(null)	40	35
11	19	LLuro	Javier	07-SEP-20	04235070	(null)	100	35
12	20	Fernan	Rofrigo	15-APR-21	04215899	(null)	100	40
13	22	Fernan	Jimena	15-APR-21	04215792	(null)	50	45
14	23	Garcia	Lopez	17-APR-21	04215891	(null)	40	45
15	24	Malfrida	Infanta	18-APR-21	04215991	0.15	100	45
16	25	Alvarez	Rofrigo	15-APR-21	04215896	(null)	40	50
17	26	Garcia	Lopez	17-APR-21	04215871	(null)	100	50
18	33	Gustav	Klauss	17-APR-21	09215867	(null)	40	70
19	34	Mahstern	Mark	17-APR-21	09315867	(null)	40	75
20	35	Friedrichson	Karl	17-APR-21	09415867	(null)	40	80
21	1009	Andrei	Mihailescu	08-MAY-19	07239800	(null)	70	10
22	0	Popescu	Mihnea	08-MAY-19	07239890	(null)	40	5
23	1001	Andrei	Mihailescu	08-MAY-19	08239890	(null)	70	10
24	1007	Andrei	Mihailescu	08-MAY-19	08239800	(null)	70	10
25	1008	Andrei	Mihailescu	08-MAY-19	07209800	(null)	70	10
26	2000	Andrei	Mihailescu	08-MAY-19	01209800	(null)	70	10

12. Definiți un triggerde tip LDD. Declanșați trigger-ul.

Cerinta: Pentru a evita posibilitatea de a perturba acticitatea comerciala, schema bazei de date se poate altera doar in prima sau ultima zi din luna. De asemenea, se retine un istoric al modificarilor efectuate asupra schemei.

```
--12 trigger LDD cu lansare de exceptii in script output cu salvare de informatii

create table istoric_modificari(

nume_user varchar(40),

nume_obiect_modificat varchar(35),

nume_comanda varchar(30),

data_modificare date);


create or replace trigger alterare_schema before create or alter or drop on schema

begin

   insert into istoric_modificari

      values(sys.login_user, sys.dictionary_obj_name, sys.sysevent, sysdate);
```

```
   if (to_char(sysdate) != to_char(last_day(sysdate)) or to_char(sysdate) != to_char(trunc(sysdate, 'mm'))) then

      RAISE_APPLICATION_ERROR(-20100, 'Se modifica schema doar in prima sau ultima zi din luna!');

   end if;

end;

/


--alter trigger alterare_schema enable;

--alter trigger alterare_schema disable;


create table testare(

camp1 number(3),

camp2 varchar(20)

);


alter table testare add camp3 date;


drop table testare;


select * from istoric_modificari;
```



	NUME_USER	NUME_OBIECT_MODIFICAT	NUME_COMANDA	DATA_MODIFICARE
1	UTILIZ	TESTARE	CREATE	13-JAN-23
2	UTILIZ	TESTARE	ALTER	13-JAN-23
3	UTILIZ	TESTARE	DROP	13-JAN-23
4	UTILIZ	SUBPROGRAME_PROIECT	CREATE	13-JAN-23
5	UTILIZ	SUBPROGRAME_PROIECT	CREATE	13-JAN-23
6	UTILIZ	NR_ANG_EXTREM_STAT	CREATE	13-JAN-23
7	UTILIZ	ESTE_LOCAL	CREATE	13-JAN-23
8	UTILIZ	SE_COMERCIALIZEAZA_IN	CREATE	13-JAN-23
9	UTILIZ	CASA_DESCHISA	CREATE	13-JAN-23
10	UTILIZ	CASA_DESCHISA	CREATE	13-JAN-23
11	UTILIZ	CASA_DESCHISA	CREATE	13-JAN-23
12	UTILIZ	ROLURI_NECESARE_MAGAZIN	CREATE	13-JAN-23
13	UTILIZ	TESTARE	CREATE	13-JAN-23
14	UTILIZ	TESTARE	ALTER	13-JAN-23
15	UTILIZ	TESTARE	DROP	13-JAN-23

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

```
--13 pachet cu procedurile si functiile de la 6 la 9

create or replace package subprograme_proiect as

   procedure display_emplpoyees_city(oras cities.city_name%type); --6


   procedure nr_ang_extrem_stat (nume_oras cities.city_name%type, selectie number); --7


   function este_local (cod markets.id_market%type) return boolean; --8


   procedure se_comercializeaza_in (furnizor in suppliers.name_s%type, tara in
countries.state_name%type, status out boolean); --9

end subprograme_proiect;

/


create or replace package body subprograme_proiect as

   --6
```

```
    procedure display_emplpoyees_city(oras cities.city_name%type) is
    type emp_rec is record( prenume employees.first_name%type,
                            nume employees.last_name%type,
                            department departments.departmentg_name%type,
                            salariu departments.base_salary%type);
    type tablou_indexat is table of emp_rec index by pls_integer;
    type vector_id_emp is varray(100) of employees.id_emp%type;
    v_cod_oras cities.city_name%type;
    v_ids vector_id_emp;
    tab_emp tablou_indexat;
begin
    select city_tag into v_cod_oras from cities where upper(oras) = upper(city_name);

    select employees.id_emp bulk collect into v_ids
    from cities join (addresses) using (city_tag)
            join markets using(address_code)
               join employees using (id_market)
    where city_tag = v_cod_oras;

    for i in v_ids.first..v_ids.last loop
      select first_name, last_name, departmentg_name, base_salary
      into tab_emp(i)
      from employees join departments using (id_depart)
      where id_emp = v_ids(i);
    end loop;

    DBMS_OUTPUT.PUT_LINE('In orasul ' || oras || ' lucreaza ' || tab_emp.count || ' angajati');

    for i in tab_emp.first..tab_emp.last loop
```

```
      DBMS_OUTPUT.PUT_LINE(tab_emp(i).prenume || ' ' || tab_emp(i).nume || ', cu salariul de baza ' ||
tab_emp(i).salariu || ' la departamentul ' || tab_emp(i).department);
   end loop;

exception
   when TOO_MANY_ROWS then DBMS_OUTPUT.PUT_LINE('Mai multe orase cu numele ' || oras || '!');
   when NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Orasul ' || oras || ' nu exista!');
   when OTHERS then DBMS_OUTPUT.PUT_LINE('Eroare necunoscuta!');
end display_emplpoyees_city;

   --7
   procedure nr_ang_extrem_stat (nume_oras cities.city_name%type, selectie number) is
   subquery exception;
   pragma EXCEPTION_INIT(subquery, -01427);
   cursor c_magazine (nume_tara countries.state_name%type) is
      select id_market, count (*) nr_ang
      from (select id_market, id_emp
            from countries co
               join cities ci on (ci.state_tag = co.state_tag and upper(co.state_name) = upper(nume_tara))
               join addresses using (city_tag)
               join markets using (address_code)
               join employees using(id_market))
      group by id_market;
   cursor c_suprafata is
      select *
      from markets
      where surface >= selectie;
   type tab_indexat is table of markets%rowtype index by pls_integer;
   v_state_name countries.state_name%type;
```

```
    v_id_market markets.id_market%type;
    v_surface markets.surface%type;
    v_market markets%rowtype;
    v_index number := 1;
    v_tab tab_indexat;
begin
    select state_name into v_state_name from countries
    where (select state_tag
          from cities
          where upper(city_name) = upper(nume_oras)) = state_tag;

    if selectie > 0 then
       open c_suprafata;

       loop
          fetch c_suprafata into v_tab(v_index);
          exit when c_suprafata%NOTFOUND;
          v_index := v_index + 1;
       end loop;

       close c_suprafata;

       for rec in c_magazine(v_state_name) loop
          for i in v_tab.first..v_tab.last loop
             if v_tab(i).id_market = rec.id_market
                then DBMS_OUTPUT.PUT_LINE('Magazinul cu id-ul ' || rec.id_market || ' si ' || rec.nr_ang ||
' angajati are suprafata de ' || v_tab(i).surface || ' mp');
             end if;
          end loop;
```

```
      end loop;
   else
      DBMS_OUTPUT.PUT_LINE('Optiune incorecta pentru selectia suprafetei');
   end if;
exception
   when NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Orasul nu exista!');
   when subquery then DBMS_OUTPUT.PUT_LINE('Mai multe orase cu acest nume!');
   when OTHERS then DBMS_OUTPUT.PUT_LINE('Eroare necunoscuta!');
end nr_ang_extrem_stat;


   --8
   function este_local (cod markets.id_market%type) return boolean is
   fara_return exception;
   cursor c_nr_prod is
   select id_market, nvl(nr_prod,0) nr_prod
   from (select id_market, count(*) nr_prod
        from supply
        group by id_market) full outer join markets using(id_market)
   order by id_market;
   type prod_rec is record(id markets.id_market%type,
                           nr_prod number);
   type tablou_indexat is table of prod_rec index by pls_integer;
   type tablou_imbricat is table of prod_rec;
   v_locale tablou_indexat;
   v_totale tablou_imbricat := tablou_imbricat();
begin
   select id_market, nvl(nr_prod,0)
   bulk collect into v_locale
   from   (select id_market, count(*) nr_prod
```

```
        from supply
        join suppliers using (id_supplier)
        join markets using (id_market)
        join addresses using (address_code)
        join cities using (city_tag)
        join countries using (state_tag)
        where org_state = state_tag
        group by id_market)
        full outer join markets using(id_market)
order by id_market;

open c_nr_prod;

loop
    v_totale.extend;
    fetch c_nr_prod into v_totale(v_totale.last);
    exit when c_nr_prod%NOTFOUND;
end loop;

close c_nr_prod;

for i in v_locale.first..v_locale.last loop
    if (v_locale(i).id = cod) then
        if (v_locale(i).nr_prod / v_totale(i).nr_prod > 0.5) then
            return true;
        else
            if (v_locale(i).nr_prod / v_totale(i).nr_prod <= 0.5) then
                return false;
            end if;
```

```
        end if;
      end if;
   end loop;

   raise fara_return;

exception
   --when NO_DATA_FOUND
   --    then DBMS_OUTPUT.PUT_LINE('Introduceti un cod valid!');
   --    return null;
   when ZERO_DIVIDE
      then DBMS_OUTPUT.PUT_LINE('Magazinul inca nu este aprovizionat!');
      return null;
   when fara_return
      then DBMS_OUTPUT.PUT_LINE('Introduceti un cod valid!');
      return null;
   when OTHERS
      then DBMS_OUTPUT.PUT_LINE('Eroare necunoscuta!');
      return null;
   --cazul TOO_MANY_ROWS nu are sens deoarece parametrul este un id
   --cazul NO_DATA_FOUND nu are sens deoarece, din structura functiei,
   --un cod care nu e valid genereaza un cursor gol deci se ajunge la final
   --fara sa se returneze ceva, nu se intra in cele doua foruri
end este_local;

   --9
   procedure se_comercializeaza_in (furnizor in suppliers.name_s%type, tara in countries.state_name%type, status out boolean) is
   --definim exceptii noi ca sa tratam cazurile de exceptie pentru fiecare parametru
```

```
    no_furnizor exception;
    too_many_furnizor exception;
    no_stat exception;
    too_many_stat exception;
    v_exista number := 0;
    v_nr_rez number;
begin
    select count (*) into v_nr_rez
        from (select state_tag
              from countries
              where upper(state_name) = upper(tara));

    if (v_nr_rez > 1)
        then raise too_many_stat;
    end if;

    if (v_nr_rez < 1)
        then raise no_stat;
    end if;

    select count (*) into v_nr_rez
        from (select id_supplier
              from suppliers
              where upper(name_s) = upper(furnizor));

    if (v_nr_rez > 1)
        then raise too_many_furnizor;
    end if;
```

```
    if (v_nr_rez < 1)
        then raise no_furnizor;
    end if;

    for i in   (select id_supplier--, count(id_supplier) nr_prod
                from supply
                join suppliers using (id_supplier)
                join markets using (id_market)
                join addresses using (address_code)
                join cities using (city_tag)
                join countries using (state_tag)
                where (upper(tara) = upper(state_name)
                and  upper(name_s) = upper(furnizor))
                --group by id_supplier
                order by state_tag) loop
        status := true;
        v_exista := v_exista + 1;
    end loop;

    if v_exista = 0 then
        status := false;
        DBMS_OUTPUT.PUT_LINE('Furnizorul nu are produse pe piata!');
    else
        DBMS_OUTPUT.PUT_LINE('Furnizorul are produse pe piata!');
    end if;

exception
    when no_furnizor then
        DBMS_OUTPUT.PUT_LINE('Furnizor inexistent!');
```

```
        status := null;
    when too_many_furnizor then
        DBMS_OUTPUT.PUT_LINE('Mai multi furnizori cu acelasi nume!');
        status := null;
    when no_stat then
        DBMS_OUTPUT.PUT_LINE('Stat inexistent!');
        status := null;
    when too_many_stat then
        DBMS_OUTPUT.PUT_LINE('Mai multe state cu acelasi nume!');
        status := null;
    --when others then
    --    DBMS_OUTPUT.PUT_LINE('Eroare neidentificata!');
    --    status := null;
end se_comercializeaza_in;
end subprograme_proiect;
/

begin
    subprograme_proiect.display_emplpoyees_city('Bucharest');
end;
/

begin
    subprograme_proiect.nr_ang_extrem_stat('Bucharest', 100);
end;
/

declare
    v_status boolean;
```

```
begin
   v_status := subprograme_proiect.este_local(15); --local

   if (v_status = true) then
      DBMS_OUTPUT.PUT_LINE('Magazin local');
   else
      DBMS_OUTPUT.PUT_LINE('Magazin international');
   end if;
end;
/

declare
   v_status boolean;
begin
   v_status := subprograme_proiect.este_local(10); --international

   if (v_status = true) then
      DBMS_OUTPUT.PUT_LINE('Magazin local');
   end if;
   if (v_status = false) then
      DBMS_OUTPUT.PUT_LINE('Magazin international');
   end if;
end;
/

declare
   este boolean;
begin
   subprograme_proiect.se_comercializeaza_in('Ferma lui Ion', 'Romania', este); -- da
```

```
   if este = true then
      DBMS_OUTPUT.PUT_LINE('Confirmare, se comercializeaza!');
   end if;

   if este = false then
      DBMS_OUTPUT.PUT_LINE('Confirmare, nu se comercializeaza!');
   end if;
end;
/

declare
   este boolean;
begin
   subprograme_proiect.se_comercializeaza_in('Ferma lui Ion', 'Spain', este); -- nu

   if este = true then
      DBMS_OUTPUT.PUT_LINE('Confirmare, se comercializeaza!');
   end if;

   if este = false then
      DBMS_OUTPUT.PUT_LINE('Confirmare, nu se comercializeaza!');
   end if;
end;
/


--10 trigger LMD la nivel de comanda cu lansare de exceptii in script output
create or replace trigger casa_deschisa before delete or update on products
```

```
begin
   if to_char(sysdate, 'day') = 'SUNDAY' THEN
      if to_number(to_char(sysdate, 'hh24')) < 11 then
         RAISE_APPLICATION_ERROR(-21001, 'Magazinele se deschid dupa ora 11:00 duminica!');
      end if;

      if to_number(to_char(sysdate, 'hh24')) >= 19 then
         RAISE_APPLICATION_ERROR(-21002, 'magazinele se inchid dupa ora 19:00 in weekend!');
      end if;

   else
      if to_number(to_char(sysdate, 'hh24')) < 10 then
         RAISE_APPLICATION_ERROR(-21003, 'Magazinele se deschid dupa ora 10:00 de luni pana
sambata!');
      end if;

      if to_number(to_char(sysdate, 'hh24')) >= 21 then
         RAISE_APPLICATION_ERROR(-21004, 'Magazinele se inchid dupa ora 21:00 de luni pana sambata!');
      end if;

   end if;
end;
/

begin
   update products
   set price = price + price * 0.10
   where prod_name = 'hand cream';
end;
```

```
/

select * from products;

rollback;

begin
   delete from products
   where prod_name = 'beef';
end;
/

select * from products;

rollback;

--11 trigger LMD la nivel de linie cu lansare de exceptii in script output
create or replace trigger roluri_necesare_magazin before insert on employees for each row
declare
   v_nr_sales number;
   v_nr_board number;
   v_id_old_depart departments.id_depart%type;
   v_id_sales departments.id_depart%type;
   v_id_board departments.id_depart%type;
   sales_not_enough_insert exception;
   board_not_enough_insert exception;
begin
   select id_depart
   into v_id_sales
```

```
from departments
where upper(departmentg_name) = upper('Sales');

select id_depart
into v_id_board
from departments
where upper(departmentg_name) = upper('Board');

select count(*)
into v_nr_sales
from markets
join employees using(id_market)
join departments using(id_depart)
where id_market = :new.id_market
and id_depart = v_id_sales;

select count(*)
into v_nr_board
from markets
join employees using(id_market)
join departments using(id_depart)
where id_market = :new.id_market
and id_depart = v_id_board;

if (v_nr_sales < 1 and :new.id_depart != v_id_sales) then
    RAISE_APPLICATION_ERROR(-20001, 'Angajati mai intai un vanzator!');
end if;

if (v_nr_board < 1 and :new.id_depart != v_id_board) then
```

```
        if (v_nr_sales > 0 and :new.id_depart = v_id_sales) then
            RAISE_APPLICATION_ERROR(-20002, 'Angajati mai intai un manager!');
        end if;
    end if;
end;
/

begin
    insert into employees   --mai intai manager
        values(1000,'Andrei','Mihailescu','08-May-2019','08239890',null,40,5);
end;
/

begin
    insert into employees   --mai intai un vanzator
        values(emp_seq.nextval,'Klogge','Frau','17-Apr-2021','09515867',null,80,85);
end;
/

begin
    insert into employees   --inserare fara erori
        values(1200,'Andrei','Mihailescu','08-May-2019','01209800',null,70,10);
end;
/

select * from employees;

rollback;
```

```
--12 trigger LDD cu lansare de exceptii in script output cu salvare de informatii
create table istoric_modificari(
nume_user varchar(40),
nume_obiect_modificat varchar(35),
nume_comanda varchar(30),
data_modificare date);

create or replace trigger alterare_schema before create or alter or drop on schema
begin
   insert into istoric_modificari
      values(sys.login_user, sys.dictionary_obj_name, sys.sysevent, sysdate);

   if (to_char(sysdate) = to_char(last_day(sysdate)) or to_char(sysdate) = to_char(trunc(sysdate, 'mm')))
then
      RAISE_APPLICATION_ERROR(-20100, 'Se modifica schema doar in prima sau ultima zi din luna!');
   end if;
end;
/

create table testare(
camp1 number(3),
camp2 varchar(20)
);

alter table testare add camp3 date;

drop table testare;

select * from istoric_modificari;
```

```
--13 pachet cu procedurile si functiile de la 6 la 9
create or replace package subprograme_proiect as
   procedure display_emplpoyees_city(oras cities.city_name%type); --6

   procedure nr_ang_extrem_stat (nume_oras cities.city_name%type, selectie number); --7

   function este_local (cod markets.id_market%type) return boolean; --8

   procedure se_comercializeaza_in (furnizor in suppliers.name_s%type, tara in
countries.state_name%type, status out boolean); --9
end subprograme_proiect;
/

create or replace package body subprograme_proiect as
   --6
   procedure display_emplpoyees_city(oras cities.city_name%type) is
   type emp_rec is record( prenume employees.first_name%type,
                    nume employees.last_name%type,
                    department departments.departmentg_name%type,
                    salariu departments.base_salary%type);
   type tablou_indexat is table of emp_rec index by pls_integer;
   type vector_id_emp is varray(100) of employees.id_emp%type;
   v_cod_oras cities.city_name%type;
   v_ids vector_id_emp;
   tab_emp tablou_indexat;
begin
   select city_tag into v_cod_oras from cities where upper(oras) = upper(city_name);
```

```
   select employees.id_emp bulk collect into v_ids
   from cities join (addresses) using (city_tag)
            join markets using(address_code)
               join employees using (id_market)
   where city_tag = v_cod_oras;

   for i in v_ids.first..v_ids.last loop
     select first_name, last_name, departmentg_name, base_salary
     into tab_emp(i)
     from employees join departments using (id_depart)
     where id_emp = v_ids(i);
   end loop;

   DBMS_OUTPUT.PUT_LINE('In orasul ' || oras || ' lucreaza ' || tab_emp.count || ' angajati');

   for i in tab_emp.first..tab_emp.last loop
      DBMS_OUTPUT.PUT_LINE(tab_emp(i).prenume || ' ' || tab_emp(i).nume || ', cu salariul de baza ' ||
tab_emp(i).salariu || ' la departamentul ' || tab_emp(i).department);
   end loop;

exception
   when TOO_MANY_ROWS then DBMS_OUTPUT.PUT_LINE('Mai multe orase cu numele ' || oras || '!');
   when NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Orasul ' || oras || ' nu exista!');
   when OTHERS then DBMS_OUTPUT.PUT_LINE('Eroare necunoscuta!');
end display_emplpoyees_city;

   --7
   procedure nr_ang_extrem_stat (nume_oras cities.city_name%type, selectie number) is
   subquery exception;
```

```
    pragma EXCEPTION_INIT(subquery, -01427);
    cursor c_magazine (nume_tara countries.state_name%type) is
        select id_market, count (*) nr_ang
        from (select id_market, id_emp
            from countries co
                join cities ci on (ci.state_tag = co.state_tag and upper(co.state_name) = upper(nume_tara))
                join addresses using (city_tag)
                join markets using (address_code)
                join employees using(id_market))
        group by id_market;
    cursor c_suprafata is
       select *
       from markets
       where surface >= selectie;
    type tab_indexat is table of markets%rowtype index by pls_integer;
    v_state_name countries.state_name%type;
    v_id_market markets.id_market%type;
    v_surface markets.surface%type;
    v_market markets%rowtype;
    v_index number := 1;
    v_tab tab_indexat;
begin
    select state_name into v_state_name from countries
    where (select state_tag
          from cities
          where upper(city_name) = upper(nume_oras)) = state_tag;

    if selectie > 0 then
       open c_suprafata;
```

```
        loop
            fetch c_suprafata into v_tab(v_index);
            exit when c_suprafata%NOTFOUND;
            v_index := v_index + 1;
        end loop;

        close c_suprafata;

        for rec in c_magazine(v_state_name) loop
            for i in v_tab.first..v_tab.last loop
                if v_tab(i).id_market = rec.id_market
                    then DBMS_OUTPUT.PUT_LINE('Magazinul cu id-ul ' || rec.id_market || ' si ' || rec.nr_ang ||
' angajati are suprafata de ' || v_tab(i).surface || ' mp');
                end if;
            end loop;
        end loop;
    else
        DBMS_OUTPUT.PUT_LINE('Optiune incorecta pentru selectia suprafetei');
    end if;
exception
    when NO_DATA_FOUND then DBMS_OUTPUT.PUT_LINE('Orasul nu exista!');
    when subquery then DBMS_OUTPUT.PUT_LINE('Mai multe orase cu acest nume!');
    when OTHERS then DBMS_OUTPUT.PUT_LINE('Eroare necunoscuta!');
end nr_ang_extrem_stat;

    --8
    function este_local (cod markets.id_market%type) return boolean is
    fara_return exception;
```

```
   cursor c_nr_prod is
   select id_market, nvl(nr_prod,0) nr_prod
   from (select id_market, count(*) nr_prod
        from supply
        group by id_market) full outer join markets using(id_market)
   order by id_market;
   type prod_rec is record(id markets.id_market%type,
                           nr_prod number);
   type tablou_indexat is table of prod_rec index by pls_integer;
   type tablou_imbricat is table of prod_rec;
   v_locale tablou_indexat;
   v_totale tablou_imbricat := tablou_imbricat();
begin
   select id_market, nvl(nr_prod,0)
   bulk collect into v_locale
   from   (select id_market, count(*) nr_prod
           from supply
           join suppliers using (id_supplier)
           join markets using (id_market)
           join addresses using (address_code)
           join cities using (city_tag)
           join countries using (state_tag)
           where org_state = state_tag
           group by id_market)
           full outer join markets using(id_market)
   order by id_market;

   open c_nr_prod;
```

```
    loop
        v_totale.extend;
        fetch c_nr_prod into v_totale(v_totale.last);
        exit when c_nr_prod%NOTFOUND;
    end loop;

    close c_nr_prod;

    for i in v_locale.first..v_locale.last loop
        if (v_locale(i).id = cod) then
            if (v_locale(i).nr_prod / v_totale(i).nr_prod > 0.5) then
                return true;
            else
                if (v_locale(i).nr_prod / v_totale(i).nr_prod <= 0.5) then
                    return false;
                end if;
            end if;
        end if;
    end loop;

    raise fara_return;

exception
    --when NO_DATA_FOUND
    --    then DBMS_OUTPUT.PUT_LINE('Introduceti un cod valid!');
    --    return null;
    when ZERO_DIVIDE
        then DBMS_OUTPUT.PUT_LINE('Magazinul inca nu este aprovizionat!');
        return null;
```

```
    when fara_return
        then DBMS_OUTPUT.PUT_LINE('Introduceti un cod valid!');
        return null;
    when OTHERS
        then DBMS_OUTPUT.PUT_LINE('Eroare necunoscuta!');
        return null;
    --cazul TOO_MANY_ROWS nu are sens deoarece parametrul este un id
    --cazul NO_DATA_FOUND nu are sens deoarece, din structura functiei,
    --un cod care nu e valid genereaza un cursor gol deci se ajunge la final
    --fara sa se returneze ceva, nu se intra in cele doua foruri
end este_local;


    --9
    procedure se_comercializeaza_in (furnizor in suppliers.name_s%type, tara in
countries.state_name%type, status out boolean) is
    --definim exceptii noi ca sa tratam cazurile de exceptie pentru fiecare parametru
    no_furnizor exception;
    too_many_furnizor exception;
    no_stat exception;
    too_many_stat exception;
    v_exista number := 0;
    v_nr_rez number;
begin
    select count (*) into v_nr_rez
        from (select state_tag
            from countries
            where upper(state_name) = upper(tara));


    if (v_nr_rez > 1)
```

```
   then raise too_many_stat;
end if;

if (v_nr_rez < 1)
   then raise no_stat;
end if;

select count (*) into v_nr_rez
   from (select id_supplier
         from suppliers
         where upper(name_s) = upper(furnizor));

if (v_nr_rez > 1)
   then raise too_many_furnizor;
end if;

if (v_nr_rez < 1)
   then raise no_furnizor;
end if;

for i in   (select id_supplier--, count(id_supplier) nr_prod
           from supply
           join suppliers using (id_supplier)
           join markets using (id_market)
           join addresses using (address_code)
           join cities using (city_tag)
           join countries using (state_tag)
           where (upper(tara) = upper(state_name)
           and  upper(name_s) = upper(furnizor))
```

```
            --group by id_supplier
            order by state_tag) loop
        status := true;
        v_exista := v_exista + 1;
    end loop;

    if v_exista = 0 then
        status := false;
        DBMS_OUTPUT.PUT_LINE('Furnizorul nu are produse pe piata!');
    else
        DBMS_OUTPUT.PUT_LINE('Furnizorul are produse pe piata!');
    end if;

exception
    when no_furnizor then
        DBMS_OUTPUT.PUT_LINE('Furnizor inexistent!');
        status := null;
    when too_many_furnizor then
        DBMS_OUTPUT.PUT_LINE('Mai multi furnizori cu acelasi nume!');
        status := null;
    when no_stat then
        DBMS_OUTPUT.PUT_LINE('Stat inexistent!');
        status := null;
    when too_many_stat then
        DBMS_OUTPUT.PUT_LINE('Mai multe state cu acelasi nume!');
        status := null;
    --when others then
    --    DBMS_OUTPUT.PUT_LINE('Eroare neidentificata!');
    --    status := null;
```

```
end se_comercializeaza_in;

end subprograme_proiect;

/

begin

   subprograme_proiect.display_emplpoyees_city('Bucharest');

end;

/
```

```
In orasul Bucharest lucreaza 9 angajati
Mihailescu Andrei, cu salariul de baza 2200 la departamentul Grocery
Mihailescu Andrei, cu salariul de baza 2200 la departamentul Grocery
Mihailescu Andrei, cu salariul de baza 2200 la departamentul Grocery
Mihailescu Andrei, cu salariul de baza 2200 la departamentul Grocery
Georgeta Amirunei, cu salariul de baza 5000 la departamentul Board
Horatiu Miron, cu salariul de baza 2200 la departamentul Grocery
Alina Ionescu, cu salariul de baza 2300 la departamentul Sales
Alexandru Popescu, cu salariul de baza 2300 la departamentul Sales
Mihnea Popescu, cu salariul de baza 2300 la departamentul Sales


PL/SQL procedure successfully completed.
```

```
begin

   subprograme_proiect.nr_ang_extrem_stat('Bucharest', 100);

end;

/
```

```
Magazinul cu id-ul 5 si 1 angajati are suprafata de 200 mp
Magazinul cu id-ul 10 si 8 angajati are suprafata de 200 mp
Magazinul cu id-ul 15 si 2 angajati are suprafata de 300 mp


PL/SQL procedure successfully completed.
```

```
declare

   v_status boolean;

begin

   v_status := subprograme_proiect.este_local(15); --local
```

```
   if (v_status = true) then
      DBMS_OUTPUT.PUT_LINE('Magazin local');
   else
      DBMS_OUTPUT.PUT_LINE('Magazin international');
   end if;
end;
/
```

```
Magazin local

PL/SQL procedure successfully completed.
```

```
declare
   v_status boolean;
begin
   v_status := subprograme_proiect.este_local(10); --international

   if (v_status = true) then
      DBMS_OUTPUT.PUT_LINE('Magazin local');
   end if;
   if (v_status = false) then
      DBMS_OUTPUT.PUT_LINE('Magazin international');
   end if;
end;
/
```

```
Magazin international

PL/SQL procedure successfully completed.
```

```
declare
```

```
   este boolean;
begin
   subprograme_proiect.se_comercializeaza_in('Ferma lui Ion', 'Romania', este); -- da

   if este = true then
      DBMS_OUTPUT.PUT_LINE('Confirmare, se comercializeaza!');
   end if;

   if este = false then
      DBMS_OUTPUT.PUT_LINE('Confirmare, nu se comercializeaza!');
   end if;
end;
/
```

```
Furnizorul are produse pe piata!
Confirmare, se comercializeaza!

PL/SQL procedure successfully completed.
```

```
declare
   este boolean;
begin
   subprograme_proiect.se_comercializeaza_in('Ferma lui Ion', 'Spain', este); -- nu

   if este = true then
      DBMS_OUTPUT.PUT_LINE('Confirmare, se comercializeaza!');
   end if;

   if este = false then
      DBMS_OUTPUT.PUT_LINE('Confirmare, nu se comercializeaza!');
   end if;
```

```
end;
/
```

```
Furnizorul nu are produse pe piata!
Confirmare, nu se comercializeaza!


PL/SQL procedure successfully completed.
```