Table of Contents

[Modelul Real 2](#_Toc155921185)

[Diagrama ER 3](#_Toc155921186)

[Diagrama Conceptuală 4](#_Toc155921187)

[Implementarea diagramei conceptuale 5](#_Toc155921188)

[Adăugare informații în tabele 8](#_Toc155921189)

[Subprogram stocat – top 3 comenzi riscante 21](#_Toc155921190)

[Subprogram stocat – angajați incomplet antrenați 26](#_Toc155921191)

[Subprogram stocat – găsește cel mai profitabil casier 29](#_Toc155921192)

[Subprogram stocat – generare meniu 35](#_Toc155921193)

[Trigger – monitorizare modificări 42](#_Toc155921194)

[Trigger – max angajați 49](#_Toc155921195)

[Trigger – protejarea tabelelor 53](#_Toc155921196)

## Modelul Real

Compania „Pizza România” a deschis multiple restaurante în diferite orașe. Va folosi o bază de date relațională pentru a stoca informații:

1. **Restaurante și Orașe**   
   Restaurantele sunt aflate în orașe din România.
2. **Angajați**  
   Compania angajează casieri, bucătari și manageri. Restaurantele sunt responsabile de a angaja personalul de care au nevoie, deci fiecare angajat a fost intervievat de un manager, cu exceptia primului manager care a fost angajat de un administrator de la sediul companiei (nu ne interesează angajații de la sediu deoarece baza de date este pentru a gestiona restaurantele). Fiecare angajat are un salariu, determinat de salariu bază și limita de bonus a job-ului.
3. **Job-uri**

Fiecare job are un salariu de bază, însă există posibilitatea de a fi crescut, dacă angajatul câștigă un bonus, până la o limită. Niciun salariu nu ar trebui să fie mai mare decăt salariul bază plus bonusul maxim, sau mai mic decât salariul bază.

1. **Ingrediente, Rețete și Alergii**   
   Compania are un catalog de rețete.Se știe ce ingrediente are fiecare rețetă, iar pentru ingrediente se știe ce alergii pot provoca.
2. **Comenzi**

Un client poate comanda multiple rețete în aceeași comandă. Comanda va fi înregistrată în sistem de către casier. Unele comenzi sunt făcute prin telefon, și vor fi livrate la o adresă. Pentru livrări, clientul va plăti un cost pentru livrare. Deoarece livrarea se face printr-o firmă de livrare third-party, nu se va ține cont de vehiculele și șoferii implicați în livrare. O comandă poate include aceeași rețetă de orice număr de ori (i.e. clientul trebuie să poate comanda două pizze în aceeași comandă).

## Diagrama ER

CASIER

nr\_casa\_de\_marcat

BUCATAR

data\_antrenament\_

de\_siguranta

MANAGER

autorizat\_sa\_

angajeze

ANGAJAT

id\_angajat#

nume

data\_angajare

salariu

ISA

1:1(0)

ISA

1:1(0)

ISA

1:1(0)

RESTAURANT

id\_restaurant#

data\_deschidere

M(0)

1

casierul a procesat comanda

1

M(0)

comanda a fost facuta la restaurant

RETETA

id\_reteta#

nume

pret

ORAS

id\_oras#

nume

1

M

restaurantul este in oras

INGREDIENT

id\_ingredient#

nume

M

M

reteta include ingredient

1

M

angajatul lucreaza la restaurant

1(0)

M(0)

angajatul a fost angajat de manager

ALERGIE

id\_alergie#

nume

M

M(0)

ingredient provoaca alergie

M(0)

M(0)

nr

comanda include reteta

COMANDA

id\_comanda#

data

LIVRARE

adresa

cost

ISA

1:1(0)

1

M(0)

angajatul are job

JOB

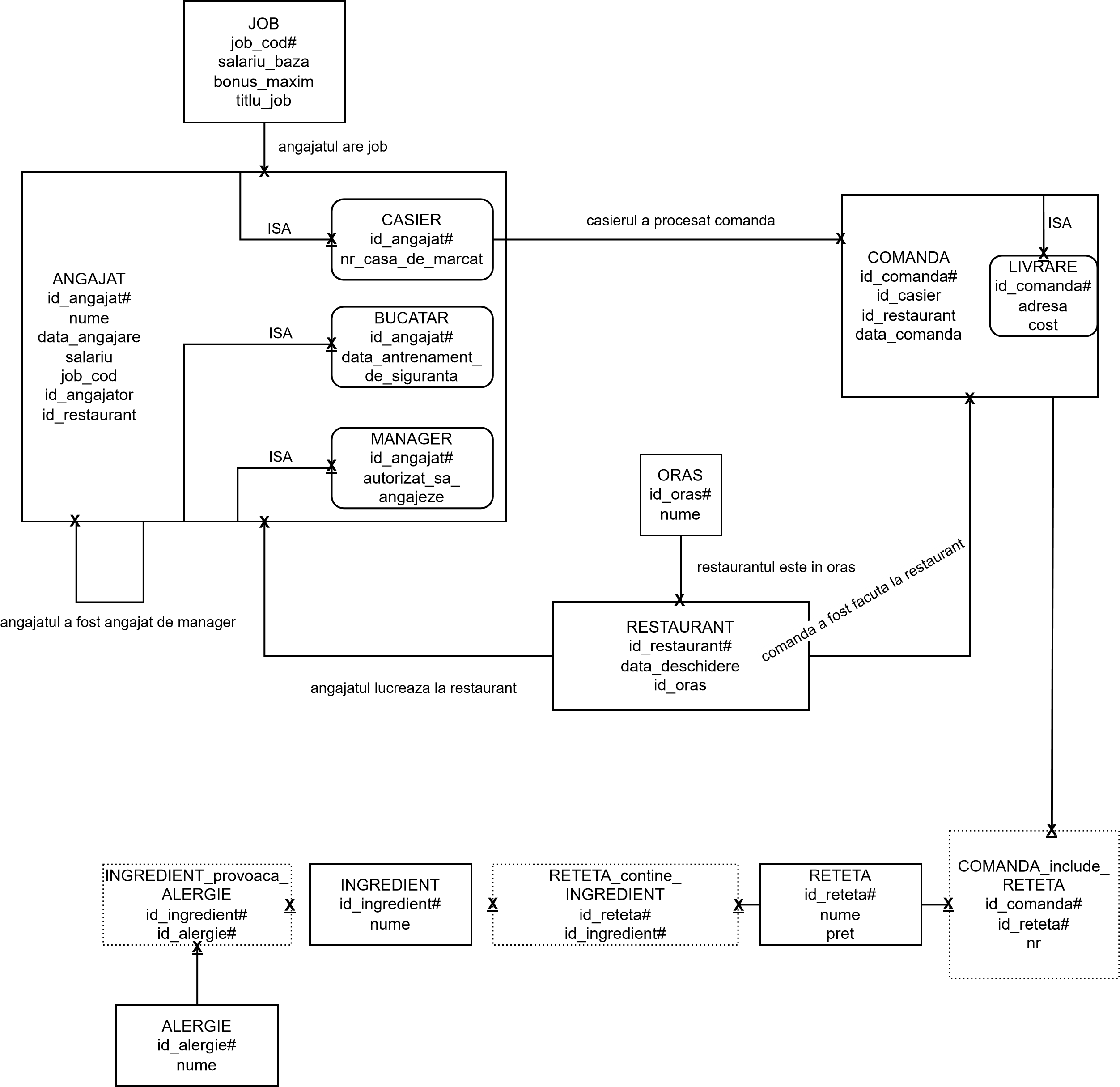
job\_cod#

salariu\_baza

bonus\_maxim

titlu\_job

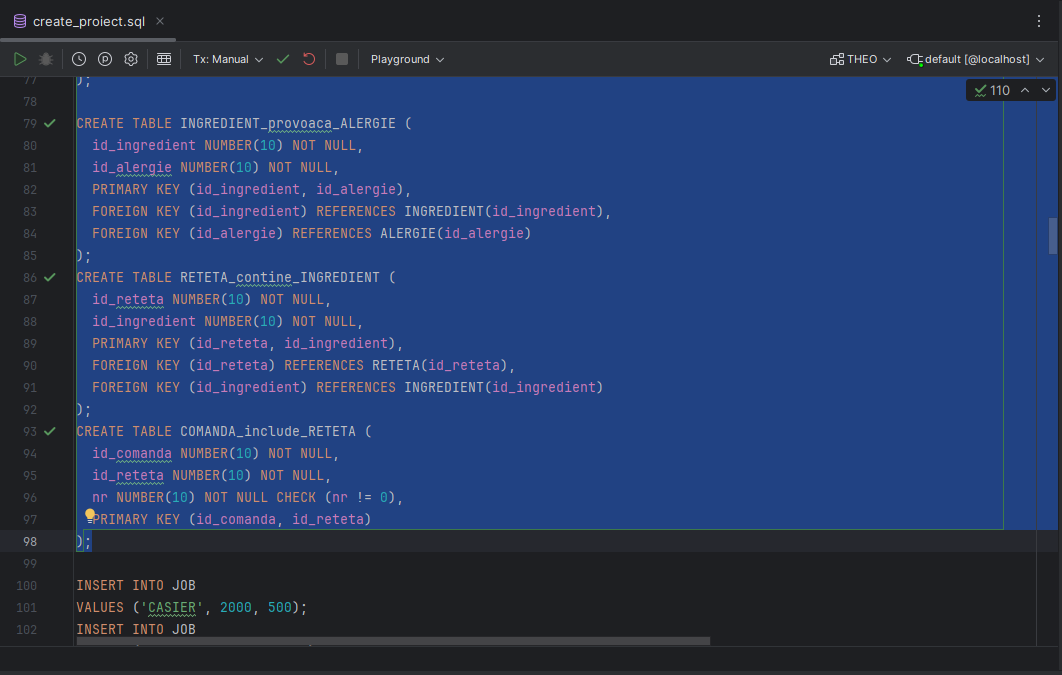
## Diagrama Conceptuală



Word nu a putut importa această diagramă ca text selectabil, ci a transformat-o în poză. Fișierul original va fi trimis cu proiectul.

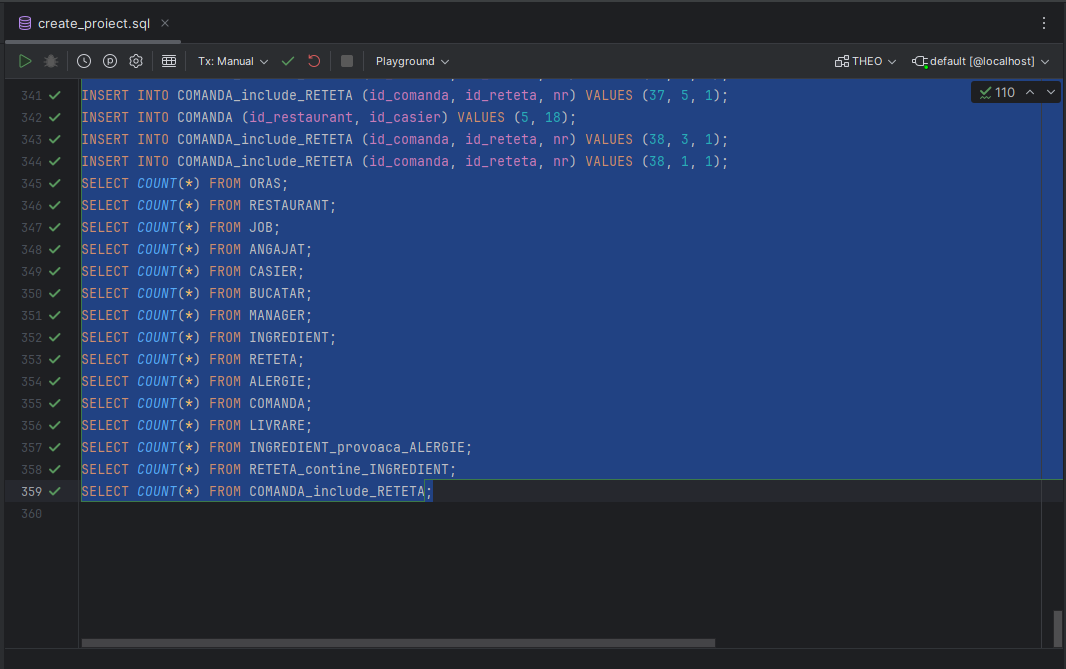
## Implementarea diagramei conceptuale

|  |
| --- |
| CREATE SEQUENCE ID\_ORAS\_SEQ START WITH 1 INCREMENT BY 1;  CREATE SEQUENCE ID\_RESTAURANT\_SEQ START WITH 1 INCREMENT BY 1;  CREATE SEQUENCE ID\_ANGAJAT\_SEQ START WITH 1 INCREMENT BY 1;  CREATE SEQUENCE ID\_INGREDIENT\_SEQ START WITH 1 INCREMENT BY 1;  CREATE SEQUENCE ID\_RETETA\_SEQ START WITH 1 INCREMENT BY 1;  CREATE SEQUENCE ID\_ALERGIE\_SEQ START WITH 1 INCREMENT BY 1;  CREATE SEQUENCE ID\_COMANDA\_SEQ START WITH 1 INCREMENT BY 1;  CREATE TABLE ORAS (    id\_oras NUMBER(10) DEFAULT ID\_ORAS\_SEQ.nextval PRIMARY KEY,    nume VARCHAR2(50) NOT NULL  );  CREATE TABLE RESTAURANT (    id\_restaurant NUMBER(10) DEFAULT ID\_RESTAURANT\_SEQ.nextval PRIMARY KEY,    id\_oras NUMBER(10) NOT NULL,    data\_deschidere DATE NOT NULL,    FOREIGN KEY (id\_oras) REFERENCES ORAS(id\_oras)  );  CREATE TABLE JOB (    job\_cod VARCHAR2(10) PRIMARY KEY,    salariu\_baza NUMBER(10) NOT NULL,    bonus\_maxim NUMBER(10) NOT NULL  );  CREATE TABLE ANGAJAT (    id\_angajat NUMBER(10) DEFAULT ID\_ANGAJAT\_SEQ.nextval PRIMARY KEY,    id\_restaurant NUMBER(10) NOT NULL,    id\_angajator NUMBER(10),    job\_cod VARCHAR2(10) NOT NULL,    nume VARCHAR2(50) NOT NULL,    data\_angajare DATE NOT NULL,    salariu NUMBER(10) NOT NULL,    FOREIGN KEY (id\_restaurant) REFERENCES RESTAURANT(id\_restaurant),    FOREIGN KEY (job\_cod) REFERENCES JOB(job\_cod)  );  CREATE TABLE CASIER (    id\_angajat NUMBER(10) PRIMARY KEY,    nr\_casa\_de\_marcat NUMBER(10),    FOREIGN KEY (id\_angajat) REFERENCES ANGAJAT(id\_angajat) ON DELETE CASCADE  );  CREATE TABLE BUCATAR (    id\_angajat NUMBER(10) PRIMARY KEY,    data\_antrenament\_de\_siguranta DATE,    FOREIGN KEY (id\_angajat) REFERENCES ANGAJAT(id\_angajat) ON DELETE CASCADE  );  CREATE TABLE MANAGER (    id\_angajat NUMBER(10) PRIMARY KEY,    autorizat\_sa\_angajeze NUMBER(1) DEFAULT 0 NOT NULL CHECK (autorizat\_sa\_angajeze = 0 OR autorizat\_sa\_angajeze = 1),    FOREIGN KEY (id\_angajat) REFERENCES ANGAJAT(id\_angajat) ON DELETE CASCADE  );  ALTER TABLE ANGAJAT ADD CONSTRAINT ANGAJATOR\_E\_MANAGER FOREIGN KEY (id\_angajator) REFERENCES MANAGER(id\_angajat);  CREATE TABLE INGREDIENT (    id\_ingredient NUMBER(10) DEFAULT ID\_INGREDIENT\_SEQ.nextval PRIMARY KEY,    nume VARCHAR2(50) NOT NULL  );  CREATE TABLE RETETA (    id\_reteta NUMBER(10) DEFAULT ID\_RETETA\_SEQ.nextval PRIMARY KEY,    nume VARCHAR2(50) NOT NULL,    pret NUMBER(10,2) NOT NULL  );  CREATE TABLE ALERGIE (    id\_alergie NUMBER(10) DEFAULT ID\_ALERGIE\_SEQ.nextval PRIMARY KEY,    nume VARCHAR2(50) NOT NULL  );  CREATE TABLE COMANDA (    id\_comanda NUMBER(10) DEFAULT ID\_COMANDA\_SEQ.nextval PRIMARY KEY,    id\_casier NUMBER(10) NOT NULL,    id\_restaurant NUMBER(10) NOT NULL,    data\_comanda DATE DEFAULT SYSDATE NOT NULL,    FOREIGN KEY (id\_casier) REFERENCES CASIER(id\_angajat),    FOREIGN KEY (id\_restaurant) REFERENCES RESTAURANT(id\_restaurant)  );  CREATE TABLE LIVRARE (    id\_comanda NUMBER(10) PRIMARY KEY,    adresa VARCHAR(100) NOT NULL,    pret NUMBER(10,2) NOT NULL,    FOREIGN KEY (id\_comanda) REFERENCES COMANDA(id\_comanda) ON DELETE CASCADE  );  CREATE TABLE INGREDIENT\_provoaca\_ALERGIE (    id\_ingredient NUMBER(10) NOT NULL,    id\_alergie NUMBER(10) NOT NULL,    PRIMARY KEY (id\_ingredient, id\_alergie),    FOREIGN KEY (id\_ingredient) REFERENCES INGREDIENT(id\_ingredient),    FOREIGN KEY (id\_alergie) REFERENCES ALERGIE(id\_alergie)  );  CREATE TABLE RETETA\_contine\_INGREDIENT (    id\_reteta NUMBER(10) NOT NULL,    id\_ingredient NUMBER(10) NOT NULL,    PRIMARY KEY (id\_reteta, id\_ingredient),    FOREIGN KEY (id\_reteta) REFERENCES RETETA(id\_reteta),    FOREIGN KEY (id\_ingredient) REFERENCES INGREDIENT(id\_ingredient)  );  CREATE TABLE COMANDA\_include\_RETETA (    id\_comanda NUMBER(10) NOT NULL,    id\_reteta NUMBER(10) NOT NULL,    nr NUMBER(10) NOT NULL CHECK (nr != 0),    PRIMARY KEY (id\_comanda, id\_reteta)  ); |



## Adăugare informații în tabele

|  |
| --- |
| INSERT INTO JOB  VALUES ('CASIER', 2000, 500);  INSERT INTO JOB  VALUES ('BUCATAR', 4000, 1000);  INSERT INTO JOB  VALUES ('MANAGER', 4500, 2500);  INSERT INTO ORAS (nume)  VALUES ('Bucuresti');  INSERT INTO ORAS (nume)  VALUES ('Cluj-Napoca');  INSERT INTO ORAS (nume)  VALUES ('Craiova');  INSERT INTO ORAS (nume)  VALUES ('Iasi');  INSERT INTO ORAS (nume)  VALUES ('Timisoara');  INSERT INTO INGREDIENT (nume) VALUES ('carne');  INSERT INTO INGREDIENT (nume) VALUES ('cartofi');  INSERT INTO INGREDIENT (nume) VALUES ('rosii');  INSERT INTO INGREDIENT (nume) VALUES ('faina');  INSERT INTO INGREDIENT (nume) VALUES ('drojdie');  INSERT INTO INGREDIENT (nume) VALUES ('sare');  INSERT INTO INGREDIENT (nume) VALUES ('aluat congelat de pizza');  INSERT INTO INGREDIENT (nume) VALUES ('ulei');  INSERT INTO ALERGIE (nume) VALUES ('gluten');  INSERT INTO ALERGIE (nume) VALUES ('rosii');  INSERT INTO ALERGIE (nume) VALUES ('drojdie');  INSERT INTO INGREDIENT\_provoaca\_ALERGIE (id\_ingredient, id\_alergie) VALUES (4, 1);  INSERT INTO INGREDIENT\_provoaca\_ALERGIE (id\_ingredient, id\_alergie) VALUES (7, 1);  INSERT INTO INGREDIENT\_provoaca\_ALERGIE (id\_ingredient, id\_alergie) VALUES (7, 3);  INSERT INTO INGREDIENT\_provoaca\_ALERGIE (id\_ingredient, id\_alergie) VALUES (3, 2);  INSERT INTO INGREDIENT\_provoaca\_ALERGIE (id\_ingredient, id\_alergie) VALUES (5, 3);  INSERT INTO RETETA (nume, pret) VALUES ('pizza', 20.99);  INSERT INTO RETETA (nume, pret) VALUES ('pizza rapida', 15.99);  INSERT INTO RETETA (nume, pret) VALUES ('cartofi prajiti', 10.99);  INSERT INTO RETETA (nume, pret) VALUES ('paine proaspata', 5.99);  INSERT INTO RETETA (nume, pret) VALUES ('cartofi prajiti cu sos', 12.99);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (1, 1);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (1, 3);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (1, 4);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (1, 5);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (1, 6);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (1, 8);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (2, 1);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (2, 3);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (2, 7);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (3, 2);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (3, 8);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (3, 6);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (4, 4);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (4, 5);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (4, 6);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (4, 8);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (5, 3);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (5, 2);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (5, 8);  INSERT INTO RETETA\_contine\_INGREDIENT (id\_reteta, id\_ingredient) VALUES (5, 6);  -- Randomly generated data  INSERT INTO RESTAURANT (id\_oras, data\_deschidere) VALUES (1, TO\_DATE('2020-01-01', 'YYYY-MM-DD'));  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (1, NULL, 'MANAGER', 'Olariu Adrian', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 5973);  INSERT INTO MANAGER (id\_angajat, autorizat\_sa\_angajeze) VALUES (1, 1);  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (1, 1, 'MANAGER', 'Murariu Corina', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 6314);  INSERT INTO MANAGER (id\_angajat, autorizat\_sa\_angajeze) VALUES (2, 0);  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (1, 1, 'CASIER', 'Iorga Irina', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 2331);  INSERT INTO CASIER (id\_angajat, nr\_casa\_de\_marcat) VALUES (3, 1);  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (1, 1, 'CASIER', 'Paduraru Alina', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 2197);  INSERT INTO CASIER (id\_angajat, nr\_casa\_de\_marcat) VALUES (4, 2);  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (1, 1, 'BUCATAR', 'Nechifor Valentin', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 4434);  INSERT INTO BUCATAR (id\_angajat, data\_antrenament\_de\_siguranta) VALUES (5, TO\_DATE('2020-01-01', 'YYYY-MM-DD'));  INSERT INTO RESTAURANT (id\_oras, data\_deschidere) VALUES (2, TO\_DATE('2020-01-01', 'YYYY-MM-DD'));  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (2, NULL, 'MANAGER', 'Lungu Ionut', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 6741);  INSERT INTO MANAGER (id\_angajat, autorizat\_sa\_angajeze) VALUES (6, 1);  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (2, 6, 'MANAGER', 'Solomon Ovidiu', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 5800);  INSERT INTO MANAGER (id\_angajat, autorizat\_sa\_angajeze) VALUES (7, 0);  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (2, 6, 'CASIER', 'Groza Anamaria', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 2143);  INSERT INTO CASIER (id\_angajat, nr\_casa\_de\_marcat) VALUES (8, 1);  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (2, 6, 'BUCATAR', 'Vaduva Radu', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 4478);  INSERT INTO BUCATAR (id\_angajat, data\_antrenament\_de\_siguranta) VALUES (9, NULL);  INSERT INTO RESTAURANT (id\_oras, data\_deschidere) VALUES (3, TO\_DATE('2020-01-01', 'YYYY-MM-DD'));  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (3, NULL, 'MANAGER', 'Tomescu Alexandru', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 6305);  INSERT INTO MANAGER (id\_angajat, autorizat\_sa\_angajeze) VALUES (10, 1);  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (3, 10, 'CASIER', 'Robu Andreea Elena', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 2266);  INSERT INTO CASIER (id\_angajat, nr\_casa\_de\_marcat) VALUES (11, 1);  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (3, 10, 'BUCATAR', 'Trandafir Ioan', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 4505);  INSERT INTO BUCATAR (id\_angajat, data\_antrenament\_de\_siguranta) VALUES (12, NULL);  INSERT INTO RESTAURANT (id\_oras, data\_deschidere) VALUES (4, TO\_DATE('2020-01-01', 'YYYY-MM-DD'));  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (4, NULL, 'MANAGER', 'Kovacs Florina', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 6369);  INSERT INTO MANAGER (id\_angajat, autorizat\_sa\_angajeze) VALUES (13, 1);  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (4, 13, 'CASIER', 'Costea Viorel', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 2360);  INSERT INTO CASIER (id\_angajat, nr\_casa\_de\_marcat) VALUES (14, 1);  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (4, 13, 'BUCATAR', 'Iordache Anamaria', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 4285);  INSERT INTO BUCATAR (id\_angajat, data\_antrenament\_de\_siguranta) VALUES (15, NULL);  INSERT INTO RESTAURANT (id\_oras, data\_deschidere) VALUES (5, TO\_DATE('2020-01-01', 'YYYY-MM-DD'));  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (5, NULL, 'MANAGER', 'Ciocan Iulia', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 5448);  INSERT INTO MANAGER (id\_angajat, autorizat\_sa\_angajeze) VALUES (16, 1);  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (5, 16, 'CASIER', 'Tomescu Iulian', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 2236);  INSERT INTO CASIER (id\_angajat, nr\_casa\_de\_marcat) VALUES (17, 1);  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (5, 16, 'CASIER', 'Vieru Iulia', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 2132);  INSERT INTO CASIER (id\_angajat, nr\_casa\_de\_marcat) VALUES (18, 2);  INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES (5, 16, 'BUCATAR', 'Dobre Claudiu', TO\_DATE('2020-01-01', 'YYYY-MM-DD'), 4937);  INSERT INTO BUCATAR (id\_angajat, data\_antrenament\_de\_siguranta) VALUES (19, NULL);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (1, 3);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (1, 2, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (1, 3, 1);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (1, 'Str. Ilie nr. 79', 17);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (1, 3);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (2, 1, 2);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (2, 'Str. Dumitrescu nr. 5 bl. 45', 18);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (1, 3);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (3, 4, 3);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (3, 3, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (3, 5, 3);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (3, 'Str. Duma nr. 54', 17);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (1, 3);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (4, 3, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (4, 1, 2);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (4, 'Str. Alexa nr. 76', 13);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (1, 3);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (5, 1, 2);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (1, 3);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (6, 2, 1);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (6, 'Str. Trandafir nr. 55 bl. 7', 12);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (1, 4);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (7, 5, 2);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (7, 'Str. Marin nr. 16 bl. 39', 19);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (1, 4);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (8, 4, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (1, 4);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (9, 1, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (9, 5, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (9, 3, 2);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (9, 4, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (1, 4);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (10, 1, 2);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (10, 'Str. Vasiliu nr. 55 bl. 17', 11);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (2, 8);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (11, 3, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (2, 8);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (12, 1, 2);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (2, 8);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (13, 5, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (2, 8);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (14, 3, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (3, 11);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (15, 2, 3);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (15, 'Str. Rotaru nr. 21', 18);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (3, 11);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (16, 1, 2);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (16, 4, 1);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (16, 'Str. Solomon nr. 88', 19);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (3, 11);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (17, 5, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (17, 3, 3);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (17, 4, 2);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (17, 2, 1);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (17, 'Str. Sandor nr. 89 bl. 30', 11);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (3, 11);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (18, 1, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (18, 3, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (3, 11);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (19, 5, 2);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (19, 3, 1);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (19, 'Str. Burlacu nr. 50', 20);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (3, 11);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (20, 1, 2);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (20, 'Str. Militaru nr. 21 bl. 15', 20);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (3, 11);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (21, 4, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (21, 5, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (3, 11);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (22, 1, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (22, 2, 2);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (22, 4, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (22, 5, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (3, 11);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (23, 5, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (23, 1, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (3, 11);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (24, 4, 3);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (24, 1, 3);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (4, 14);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (25, 1, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (25, 2, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (4, 14);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (26, 1, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (26, 2, 2);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (26, 'Str. Carp nr. 79 bl. 13', 11);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (4, 14);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (27, 2, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (27, 3, 2);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (27, 'Str. Ivan nr. 61', 20);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (4, 14);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (28, 1, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (5, 17);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (29, 3, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (29, 2, 3);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (29, 5, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (29, 4, 1);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (29, 'Str. Oprea nr. 6 bl. 44', 14);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (5, 17);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (30, 3, 2);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (30, 5, 3);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (30, 'Str. Ifrim nr. 84', 20);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (5, 18);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (31, 4, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (31, 3, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (31, 5, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (5, 18);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (32, 2, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (5, 18);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (33, 3, 2);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (33, 5, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (5, 18);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (34, 5, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (34, 1, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (34, 4, 2);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (34, 2, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (5, 18);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (35, 5, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (35, 2, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (5, 18);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (36, 2, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (36, 1, 3);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (36, 5, 2);  INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES (36, 'Str. Ciocan nr. 52', 14);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (5, 18);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (37, 1, 2);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (37, 5, 1);  INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES (5, 18);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (38, 3, 1);  INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES (38, 1, 1);  SELECT COUNT(\*) FROM ORAS;  SELECT COUNT(\*) FROM RESTAURANT;  SELECT COUNT(\*) FROM JOB;  SELECT COUNT(\*) FROM ANGAJAT;  SELECT COUNT(\*) FROM CASIER;  SELECT COUNT(\*) FROM BUCATAR;  SELECT COUNT(\*) FROM MANAGER;  SELECT COUNT(\*) FROM INGREDIENT;  SELECT COUNT(\*) FROM RETETA;  SELECT COUNT(\*) FROM ALERGIE;  SELECT COUNT(\*) FROM COMANDA;  SELECT COUNT(\*) FROM LIVRARE;  SELECT COUNT(\*) FROM INGREDIENT\_provoaca\_ALERGIE;  SELECT COUNT(\*) FROM RETETA\_contine\_INGREDIENT;  SELECT COUNT(\*) FROM COMANDA\_include\_RETETA; |



Datele aleatorii au fost generate folosind un program Python. Programul are seed-ul RNG setat, deci dă rezultate deterministe.

|  |
| --- |
| import random;  def main():      random.seed(0)      salary\_rng = random.Random(0)      sql\_code = "\n\n"      # http://www.name-statistics.org/ro/prenumecomune.php      prenume = ["Ana Maria", "Alexandru", "Mihaela", "Andreea", "Elena", "Adrian", "Andrei", "Alexandra", "Mihai", "Ionut", "Cristina", "Florin", "Daniel", "Marian", "Marius", "Cristian", "Daniela", "Alina", "Maria", "Ioana", "Constantin", "Nicoleta", "Georgiana", "Mariana", "Bogdan", "Vasile", "Gabriel", "Gabriela", "Nicolae", "Gheorghe", "George", "Ioan", "Valentin", "Adriana", "Ionela", "Catalin", "Stefan", "Ion", "Florentina", "Anca", "Anamaria", "Simona", "Iulian", "Roxana", "Oana", "Irina", "Diana", "Mirela", "Iuliana", "Madalina", "Raluca", "Ionel", "Lucian", "Cosmin", "Sorin", "Loredana", "Claudia", "Monica", "Ramona", "Dumitru", "Ana", "Ciprian", "Corina", "Laura", "Vlad", "Razvan", "Radu", "Liliana", "Valentina", "Viorel", "Iulia", "Ovidiu", "Florina", "Robert", "Catalina", "Carmen", "Claudiu", "Alin", "Oana Maria", "Camelia", "Andreea Elena", "Dan", "Costel", "Alina Elena", "Elena Cristina", "Mircea", "Laurentiu", "Georgeta", "Maria Cristina", "Paul", "Alina Maria", "Dragos", "Silviu"];      # https://numedefamilie.eu/romania      nume = ["Popa", "Popescu", "Pop", "Radu", "Dumitru", "Stan", "Stoica", "Gheorghe", "Matei", "Rusu", "Mihai", "Ciobanu", "Constantin", "Marin", "Ionescu", "Florea", "Ilie", "Toma", "Stanciu", "Munteanu", "Vasile", "Oprea", "Tudor", "Sandu", "Moldovan", "Ion", "Ungureanu", "Dinu", "Andrei", "Barbu", "Serban", "Neagu", "Cristea", "Anghel", "Lazar", "Dragomir", "Enache", "Badea", "Stefan", "Vlad", "Mocanu", "Iordache", "Coman", "Cojocaru", "Grigore", "Voicu", "Dobre", "Petre", "Nagy", "Lupu", "Lungu", "Ivan", "Ene", "Preda", "Roman", "Ionita", "Iancu", "Nicolae", "Balan", "Manea", "Nistor", "Stoian", "Avram", "Pavel", "Simion", "Rus", "Iacob", "Bucur", "Luca", "Olteanu", "Filip", "Tanase", "Costea", "Craciun", "David", "Stancu", "Dumitrescu", "Marcu", "Muresan", "Diaconu", "Nedelcu", "Rotaru", "Baciu", "Szabo", "Zaharia", "Costache", "Alexandru", "Suciu", "Dan", "Anton", "Bogdan", "Rosu", "Moraru", "Toader", "Paraschiv", "Sava", "Nica", "Kovacs", "Nita", "Muntean", "Constantinescu", "Albu", "Cretu", "Calin", "Olaru", "Varga", "Georgescu", "Dragan", "Popovici", "Ardelean", "Dumitrache", "Chiriac", "Petcu", "Miron", "Dima", "Mihalache", "Zamfir", "Paun", "Marinescu", "Petrescu", "Niculae", "Ghita", "Neacsu", "Soare", "Moise", "Bratu", "Damian", "Ursu", "Croitoru", "Istrate", "Sirbu", "Pascu", "Savu", "Manole", "Dinca", "Apostol", "Micu", "Stroe", "Nitu", "Draghici", "Crisan", "Tudorache", "Cozma", "Grosu", "Rosca", "Oancea", "Ignat", "Radulescu", "Adam", "Mihaila", "Sima", "Irimia", "Molnar", "Necula", "Ciocan", "Manolache", "Balint", "Grecu", "Burlacu", "Nastase", "Macovei", "Pirvu", "Turcu", "Simon", "Kiss", "Marian", "Chirila", "Panait", "Cazacu", "Teodorescu", "Trandafir", "Militaru", "Oltean", "Stanescu", "Negru", "Farcas", "Maxim", "Toth", "Gabor", "Florescu", "Dumitrascu", "Pintilie", "Tamas", "Morar", "Visan", "Cosma", "Chirita", "Danciu", "Dogaru", "Gavrila", "Tudose", "Voinea", "Dascalu", "Moldoveanu", "Lazăr", "Pana", "Mihalcea", "Patrascu", "Negrea", "Trif", "Mircea", "Ichim", "Alexe", "Grigoras", "Costin", "Iliescu", "Bejan", "Nechita", "Mirea", "Neagoe", "Cucu", "Puiu", "Musat", "Prodan", "Banu", "Stefanescu", "Olariu", "Ispas", "Szekely", "Blaga", "Danila", "Trifan", "Gal", "Groza", "Bota", "Boboc", "Maftei", "Vaduva", "Vasilescu", "Gherman", "Szasz", "Antal", "Petrea", "Martin", "Cornea", "Ganea", "Gheorghiu", "Chivu", "Pintea", "Staicu", "Niculescu", "Tănase", "Burcea", "Solomon", "Botezatu", "Miu", "Iorga", "Sabau", "Nicola", "Duta", "Pal", "Alexa", "Cirstea", "Man", "Udrea", "Aldea", "Cojocariu", "Crăciun", "Rotariu", "Negoita", "Ciobotaru", "Paduraru", "Biro", "Leonte", "Murariu", "Covaci", "Fodor", "Pricop", "Dragu", "Diaconescu", "Bodea", "Milea", "Pasca", "Carp", "Catana", "Onofrei", "Petrache", "Busuioc", "Moga", "Codreanu", "Buzatu", "Vasiliu", "Chis", "Tomescu", "Jianu", "Dragoi", "Tataru", "Ghinea", "Alecu", "Iosif", "Sandor", "Tanasa", "Epure", "şerban", "Scarlat", "Dobrin", "Radoi", "Gheorghita", "Filimon", "Veres", "Savin", "Iordan", "Nae", "Timofte", "Buta", "Duma", "ştefan", "Călin", "Achim", "Peter", "Boca", "Mitroi", "Dumitriu", "Mazilu", "Vieru", "Bunea", "Butnaru", "Ifrim", "Cristian", "Gherasim", "Mitu", "Ardeleanu", "Nechifor", "Chira", "Feraru", "Balazs", "Cazan", "Giurgiu", "Spiridon", "Marginean", "Vintila", "Palade", "Farkas", "Tofan", "Demeter", "Scurtu", "Chelaru", "Apetrei", "Vasilache", "Gradinaru", "Nicoara", "State", "Oros", "Dicu", "Ivascu", "Timis", "Marton", "Deaconu", "Robu", "Pantea"]; # from https://numedefamilie.eu/romania      def isascii(s): return len(s) == len(s.encode())      prenume = [x for x in prenume if isascii(x)]      nume = [x for x in nume if isascii(x)]      start\_date = "TO\_DATE('2020-01-01', 'YYYY-MM-DD')"      ANGAJAT\_ID\_SEQ = 0      RESTAURANT\_ID\_SEQ = 0      COMANDA\_ID\_SEQ = 0      nr\_retete = 5      nr\_orase = 5      def insert\_random\_employee(id\_restaurant, manageri, job\_cod, base\_salary, bonus\_max) -> int:          nonlocal ANGAJAT\_ID\_SEQ          nonlocal sql\_code          id\_angajator = "NULL"          if(len(manager) != 0): id\_angajator = random.choice(manager)          ANGAJAT\_ID\_SEQ += 1          emp\_salary = base\_salary          salary\_kind = salary\_rng.randint(1, 5)          match salary\_kind:              # TODO: add some values outside the allowed range, and code to warn/generate a report of disallowed salaries.              # case 1:              #     emp\_salary = base\_salary \* 0.95              # case 2:              #     emp\_salary = base\_salary + bonus\_max \* 1.05              case \_:                  emp\_salary = base\_salary + round((salary\_rng.random() \* 0.5 + salary\_rng.random() \* 0.5) \* bonus\_max)          sql\_code += f"INSERT INTO ANGAJAT (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu) VALUES ({id\_restaurant}, {id\_angajator}, '{job\_cod}', '{random.choice(nume)} {random.choice(prenume)}', {start\_date}, {emp\_salary});\n"          return ANGAJAT\_ID\_SEQ      RESTAURANT : list[int] = []      MANAGER = {}      CASIER  = {}      BASE\_SALARY\_CASIER = 2000      MAX\_BONUS\_CASIER = 500      BASE\_SALARY\_BUCATAR = 4000      MAX\_BONUS\_BUCATAR = 1000      BASE\_SALARY\_MANAGER = 4500      MAX\_BONUS\_MANAGER = 2500      for i in range(5):          RESTAURANT\_ID\_SEQ += 1          id\_oras = i + 1          RESTAURANT.append(RESTAURANT\_ID\_SEQ)          sql\_code += f"INSERT INTO RESTAURANT (id\_oras, data\_deschidere) VALUES ({id\_oras}, {start\_date});\n"          MANAGER[RESTAURANT\_ID\_SEQ] = []          CASIER[RESTAURANT\_ID\_SEQ] = []          manager = MANAGER[RESTAURANT\_ID\_SEQ]          casier = CASIER[RESTAURANT\_ID\_SEQ]          for i in range(random.randint(1, 2)):              id\_angajat = insert\_random\_employee(RESTAURANT\_ID\_SEQ, manager, "MANAGER", BASE\_SALARY\_MANAGER, MAX\_BONUS\_MANAGER)              autorizat\_sa\_angajeze = int(i == 0)              sql\_code += f"INSERT INTO MANAGER (id\_angajat, autorizat\_sa\_angajeze) VALUES ({id\_angajat}, {autorizat\_sa\_angajeze});\n"              if autorizat\_sa\_angajeze == 1:                  manager.append(id\_angajat)          casa\_de\_marcat = 0          for i in range (random.randint(1, 3)):              id\_angajat = insert\_random\_employee(RESTAURANT\_ID\_SEQ, manager, "CASIER", BASE\_SALARY\_CASIER, MAX\_BONUS\_CASIER)              nr\_casa\_de\_marcat = "NULL"              if i == 0 or random.choice([False, True, True]):                  casa\_de\_marcat += 1                  nr\_casa\_de\_marcat = casa\_de\_marcat                  casier.append(id\_angajat)              sql\_code += f"INSERT INTO CASIER (id\_angajat, nr\_casa\_de\_marcat) VALUES ({id\_angajat}, {nr\_casa\_de\_marcat});\n"          for \_ in range (random.randint(1, 1)):              id\_angajat = insert\_random\_employee(RESTAURANT\_ID\_SEQ, manager, "BUCATAR", BASE\_SALARY\_BUCATAR, MAX\_BONUS\_BUCATAR)              data\_antrenament\_de\_siguranta = random.choice([start\_date, "NULL"])              sql\_code += f"INSERT INTO BUCATAR (id\_angajat, data\_antrenament\_de\_siguranta) VALUES ({id\_angajat}, {data\_antrenament\_de\_siguranta});\n"      def random\_comanda(id\_restaurant, id\_casier):          nonlocal COMANDA\_ID\_SEQ          nonlocal sql\_code          COMANDA\_ID\_SEQ += 1          sql\_code += f"INSERT INTO COMANDA (id\_restaurant, id\_casier) VALUES ({id\_restaurant}, {id\_casier});\n"          for reteta in random.sample(range(1, nr\_retete + 1), k=random.choice([1, 1, 2, 2, 2, 3, 4])):              nr = random.choice([1, 1, 1, 1, 2, 2, 3])              sql\_code += f'INSERT INTO COMANDA\_include\_RETETA (id\_comanda, id\_reteta, nr) VALUES ({COMANDA\_ID\_SEQ}, {reteta}, {nr});\n'          if random.choice([True, False]):              adresa = f"Str. {random.choice(nume)} nr. {random.randint(1, 100)}"              if random.choice([True, False]):                  adresa += f" bl. {random.randint(1, 50)}"              cost = random.randint(10, 20)              sql\_code += f"INSERT INTO LIVRARE (id\_comanda, adresa, pret) VALUES ({COMANDA\_ID\_SEQ}, '{adresa}', {cost});\n"      for i in RESTAURANT:          for c in CASIER[i]:              for \_ in range(random.randint(1, 10)):                  random\_comanda(i, c)      TABELE = ['ORAS', 'RESTAURANT', 'JOB', 'ANGAJAT', 'CASIER', 'BUCATAR', 'MANAGER', 'INGREDIENT', 'RETETA', 'ALERGIE', 'COMANDA', 'LIVRARE', 'INGREDIENT\_provoaca\_ALERGIE', 'RETETA\_contine\_INGREDIENT', 'COMANDA\_include\_RETETA']      for table in TABELE:          sql\_code += f"SELECT COUNT(\*) FROM {table};\n"      with open("generate\_data.sql", "w") as f:          f.write(sql\_code)  if \_\_name\_\_ == "\_\_main\_\_":      main() |

## Subprogram stocat – top 3 comenzi riscante

Gradul de risc al unei comenzi e determinat de numarul de ingrediente ce provoaca alergie.

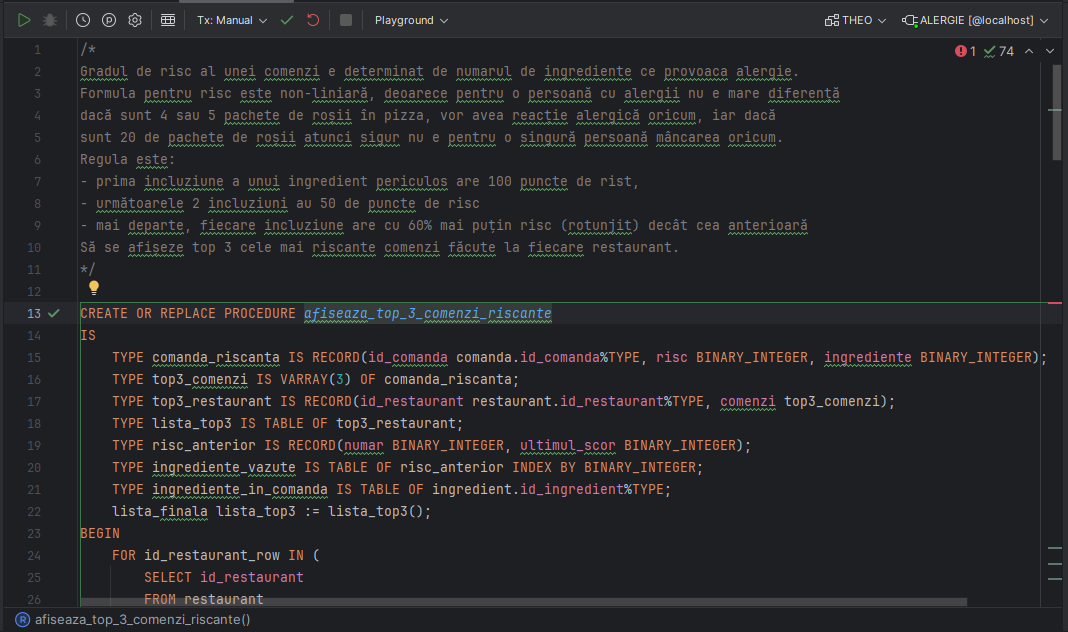
Formula pentru risc este non-liniară, deoarece pentru o persoană cu alergii nu e mare diferență dacă sunt 4 sau 5 pachete de roșii în pizza, vor avea reacție alergică oricum, iar dacă sunt 20 de pachete de roșii atunci sigur nu e pentru o singură persoană mâncarea oricum.

Regula este:

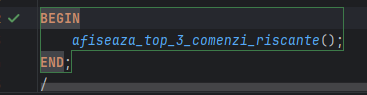
* prima incluziune a unui ingredient periculos are 100 puncte de rist,
* următoarele 2 incluziuni au 50 de puncte de risc
* mai departe, fiecare incluziune are cu 60% mai puțin risc (rotunjit) decât cea anterioară

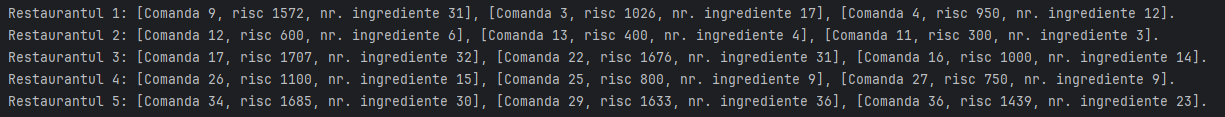
Să se afișeze top 3 cele mai riscante comenzi făcute la fiecare restaurant.

|  |
| --- |
| /\*  Gradul de risc al unei comenzi e determinat de numarul de ingrediente ce provoaca alergie.  Formula pentru risc este non-liniară, deoarece pentru o persoană cu alergii nu e mare diferență  dacă sunt 4 sau 5 pachete de roșii în pizza, vor avea reacție alergică oricum, iar dacă  sunt 20 de pachete de roșii atunci sigur nu e pentru o singură persoană mâncarea oricum.  Regula este:  - prima incluziune a unui ingredient periculos are 100 puncte de rist,  - următoarele 2 incluziuni au 50 de puncte de risc  - mai departe, fiecare incluziune are cu 60% mai puțin risc (rotunjit) decât cea anterioară  Să se afișeze top 3 cele mai riscante comenzi făcute la fiecare restaurant.  \*/  CREATE OR REPLACE PROCEDURE afiseaza\_top\_3\_comenzi\_riscante  IS      TYPE comanda\_riscanta IS RECORD(id\_comanda comanda.id\_comanda%TYPE, risc BINARY\_INTEGER, ingrediente BINARY\_INTEGER);      TYPE top3\_comenzi IS VARRAY(3) OF comanda\_riscanta;      TYPE top3\_restaurant IS RECORD(id\_restaurant restaurant.id\_restaurant%TYPE, comenzi top3\_comenzi);      TYPE lista\_top3 IS TABLE OF top3\_restaurant;      TYPE risc\_anterior IS RECORD(numar BINARY\_INTEGER, ultimul\_scor BINARY\_INTEGER);      TYPE ingrediente\_vazute IS TABLE OF risc\_anterior INDEX BY BINARY\_INTEGER;      TYPE ingrediente\_in\_comanda IS TABLE OF ingredient.id\_ingredient%TYPE;      lista\_finala lista\_top3 := lista\_top3();  BEGIN      FOR id\_restaurant\_row IN (          SELECT id\_restaurant          FROM restaurant      )      LOOP          DECLARE              blank comanda\_riscanta := comanda\_riscanta(-1, -1);              clasament top3\_comenzi := top3\_comenzi(blank, blank, blank);          BEGIN              FOR id\_comanda\_row IN (                  SELECT id\_comanda                  FROM comanda                  WHERE id\_restaurant = id\_restaurant\_row.id\_restaurant              )              LOOP                  DECLARE                      ingrediente ingrediente\_in\_comanda;                      vazute ingrediente\_vazute;                      risc\_comanda BINARY\_INTEGER := 0;                      PROCEDURE insert\_record(i NUMBER)                      IS                      BEGIN                          clasament(i) := comanda\_riscanta(id\_comanda\_row.id\_comanda, risc\_comanda, ingrediente.count);                      END;                  BEGIN                      WITH ingr AS (                          SELECT id\_ingredient, nr                          FROM comanda                          JOIN comanda\_include\_reteta                          USING (id\_comanda)                          JOIN reteta\_contine\_ingredient                          USING (id\_reteta)                          WHERE id\_comanda = id\_comanda\_row.id\_comanda                      ),                      numbers AS (                          SELECT LEVEL AS lvl                          FROM DUAL                          CONNECT BY LEVEL <= (SELECT MAX(nr) FROM ingr)                      )                      SELECT id\_ingredient                      BULK COLLECT INTO ingrediente                      FROM ingr                      JOIN numbers                      ON (nr <= lvl);                      FOR i IN 1..ingrediente.LAST                      LOOP                          DECLARE                              igr ingredient.id\_ingredient%TYPE := ingrediente(i);                          BEGIN                              IF NOT vazute.exists(ingrediente(i))                              THEN                                  vazute(igr) := risc\_anterior(0, 0);                              END IF;                              CASE vazute(igr).numar                                  WHEN 0 THEN                                      vazute(igr) := risc\_anterior(1, 100);                                  WHEN 1 THEN                                      vazute(igr) := risc\_anterior(2, 50);                                  WHEN 2 THEN                                      vazute(igr) := risc\_anterior(3, 50);                                  ELSE                                      vazute(igr) := risc\_anterior(vazute(igr).numar + 1, vazute(igr).ultimul\_scor \* 0.6);                              END CASE;                              risc\_comanda := risc\_comanda + vazute(igr).ultimul\_scor;                          END;                      END LOOP;                      IF risc\_comanda > clasament(1).risc THEN                          clasament(3) := clasament(2);                          clasament(2) := clasament(1);                          insert\_record(1);                      ELSIF risc\_comanda > clasament(2).risc THEN                          clasament(3) := clasament(2);                          clasament(2) := comanda\_riscanta(id\_comanda\_row.id\_comanda, risc\_comanda);                          insert\_record(2);                      ELSIF risc\_comanda > clasament(3).risc THEN                          insert\_record(3);                      END IF;                  END;              END LOOP;              lista\_finala.extend(1);              lista\_finala(lista\_finala.last) := top3\_restaurant(id\_restaurant\_row.id\_restaurant, clasament);          END;      END LOOP;      FOR i IN 1..lista\_finala.LAST      LOOP          DECLARE              top3 top3\_comenzi := lista\_finala(i).comenzi;              id\_rest restaurant.id\_restaurant%TYPE := lista\_finala(i).id\_restaurant;              FUNCTION str(comanda comanda\_riscanta) RETURN VARCHAR2              IS              BEGIN                  RETURN '[Comanda '||comanda.id\_comanda||', risc '||comanda.risc||', nr. ingrediente '||comanda.ingrediente||']';              END;          BEGIN              dbms\_output.PUT\_LINE(                      'Restaurantul '||id\_rest||': '||                      str(top3(1))||', '||                      str(top3(2))||', '||                      str(top3(3))||'.'              );          END;      END LOOP;  END afiseaza\_top\_3\_comenzi\_riscante;  /  BEGIN      afiseaza\_top\_3\_comenzi\_riscante();  END;  /  DROP PROCEDURE afiseaza\_top\_3\_comenzi\_riscante; |



DataGrip afișează incorect că BULK COLLECT INTO nu poate fi folosit cu WITH.



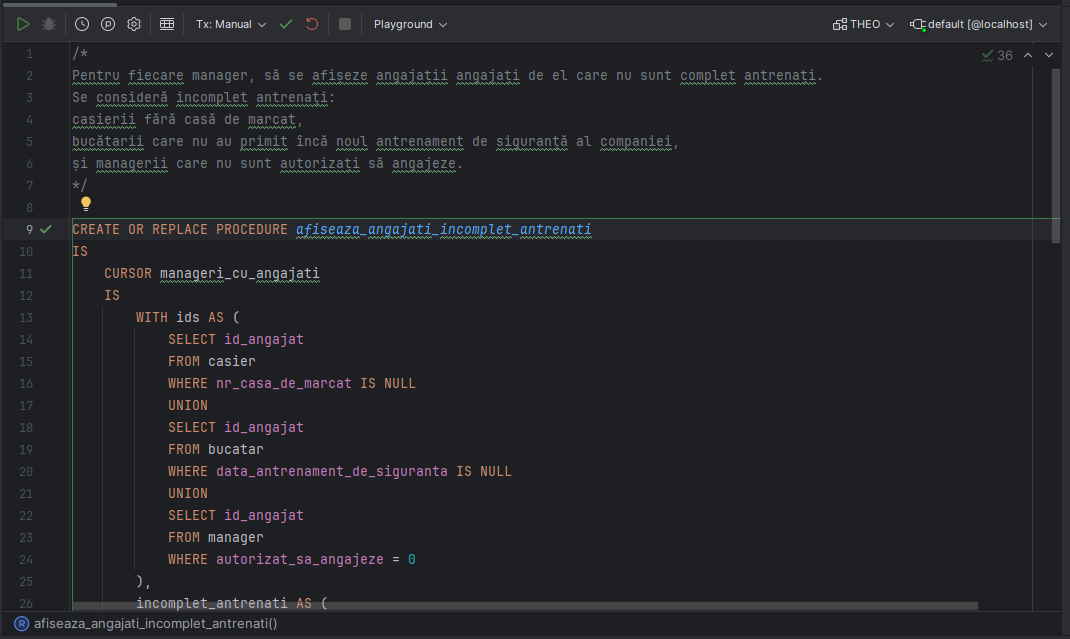


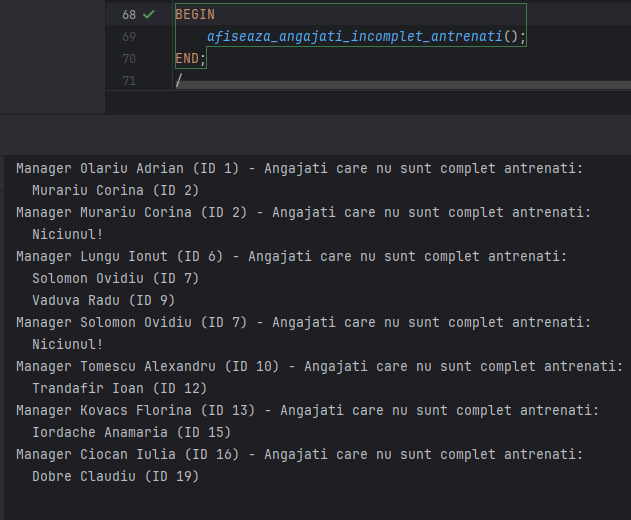
## Subprogram stocat – angajați incomplet antrenați

Pentru fiecare manager, să se afișeze angajații angajați de el care nu sunt complet antrenați. Se consideră incomplet antrenați:

* casierii fără casă de marcat,
* bucătarii care nu au primit încă noul antrenament de siguranță al companiei,
* și managerii care nu sunt autorizați să angajeze.

|  |
| --- |
| /\*  Pentru fiecare manager, să se afișeze angajații angajați de el care nu sunt complet antrenați.  Se consideră incomplet antrenați:  casierii fără casă de marcat,  bucătarii care nu au primit încă noul antrenament de siguranță al companiei,  și managerii care nu sunt autorizați să angajeze.  \*/  CREATE OR REPLACE PROCEDURE afiseaza\_angajati\_incomplet\_antrenati  IS      CURSOR manageri\_cu\_angajati      IS          WITH ids AS (              SELECT id\_angajat              FROM casier              WHERE nr\_casa\_de\_marcat IS NULL              UNION              SELECT id\_angajat              FROM bucatar              WHERE data\_antrenament\_de\_siguranta IS NULL              UNION              SELECT id\_angajat              FROM manager              WHERE autorizat\_sa\_angajeze = 0          ),          incomplet\_antrenati AS (              SELECT id\_angajat, id\_angajator, nume              FROM ids              JOIN angajat              USING (id\_angajat)          )          SELECT angajat.id\_angajat, nume, CURSOR(              SELECT id\_angajat, nume FROM incomplet\_antrenati              WHERE id\_angajator = manager.id\_angajat          ) AS angajati\_incomplet\_antrenati          FROM angajat          JOIN manager          ON angajat.id\_angajat = manager.id\_angajat;      TYPE refc IS REF CURSOR;      ref refc;      nume\_manager angajat.nume%TYPE;      id\_manager angajat.id\_angajat%TYPE;      nume\_angajat angajat.nume%TYPE;      id\_angajat angajat.id\_angajat%TYPE;  BEGIN      OPEN manageri\_cu\_angajati;      LOOP          FETCH manageri\_cu\_angajati INTO id\_manager, nume\_manager, ref;          EXIT WHEN manageri\_cu\_angajati%NOTFOUND;          dbms\_output.PUT\_LINE('Manager '||nume\_manager||' (ID '||id\_manager||') - Angajati care nu sunt complet antrenati:');          FETCH ref INTO id\_angajat, nume\_angajat;          IF ref%NOTFOUND          THEN              dbms\_output.PUT\_LINE('  Niciunul!');              CONTINUE;          END IF;          LOOP              dbms\_output.PUT\_LINE('  '||nume\_angajat||' (ID '||id\_angajat||')');              FETCH ref INTO id\_angajat, nume\_angajat;              EXIT WHEN ref%NOTFOUND;          END LOOP;          CLOSE ref;      END LOOP;      CLOSE manageri\_cu\_angajati;  END afiseaza\_angajati\_incomplet\_antrenati;  /  BEGIN      afiseaza\_angajati\_incomplet\_antrenati();  END;  /  DROP PROCEDURE afiseaza\_angajati\_incomplet\_antrenati; |





## Subprogram stocat – găsește cel mai profitabil casier

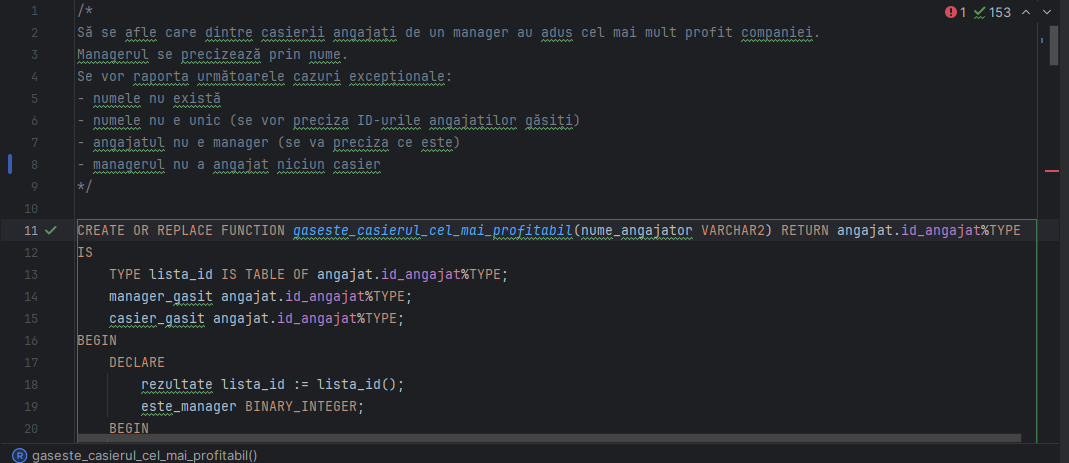
Să se afle care dintre casierii angajați de un manager au adus cel mai mult profit companiei.

Managerul se precizează prin nume.

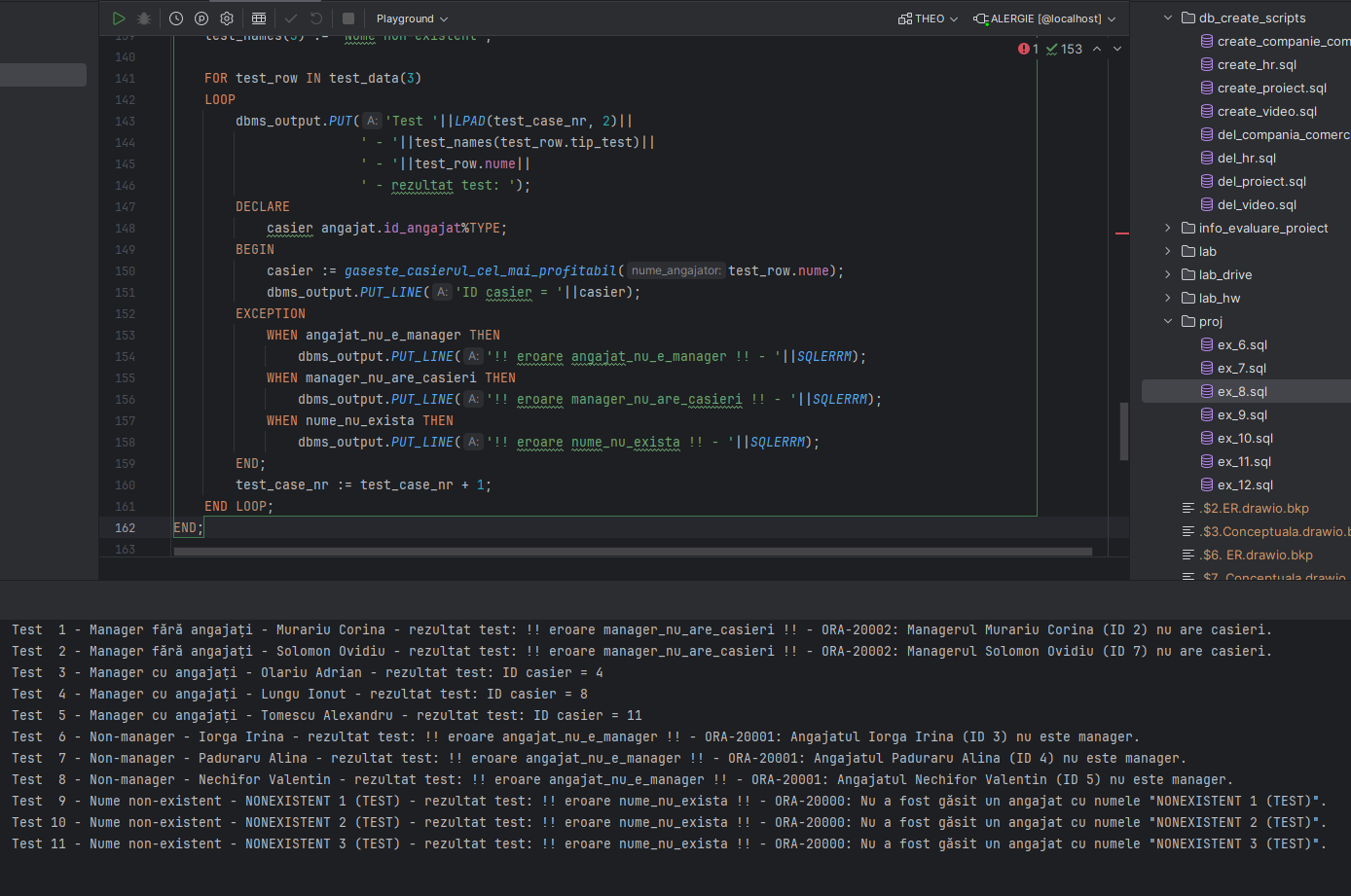
Se vor raporta următoarele cazuri excepționale:

* numele nu există
* numele nu e unic (se vor preciza ID-urile angajaților găsiți)
* angajatul nu e manager (se va preciza ce este)
* managerul nu a angajat niciun casier

|  |
| --- |
| /\*  Să se afle care dintre casierii angajați de un manager au adus cel mai mult profit companiei.  Managerul se precizează prin nume.  Se vor raporta următoarele cazuri excepționale:  - numele nu există  - numele nu e unic (se vor preciza ID-urile angajaților găsiți)  - angajatul nu e manager (se va preciza ce este)  - managerul nu a angajat niciun casier  \*/  CREATE OR REPLACE FUNCTION gaseste\_casierul\_cel\_mai\_profitabil(nume\_angajator VARCHAR2) RETURN angajat.id\_angajat%TYPE  IS      TYPE lista\_id IS TABLE OF angajat.id\_angajat%TYPE;      manager\_gasit angajat.id\_angajat%TYPE;      casier\_gasit angajat.id\_angajat%TYPE;  BEGIN      DECLARE          rezultate lista\_id := lista\_id();          este\_manager BINARY\_INTEGER;      BEGIN          SELECT id\_angajat          BULK COLLECT INTO rezultate          FROM angajat          WHERE nume = nume\_angajator;          CASE rezultate.COUNT              WHEN 0 THEN                  RAISE\_APPLICATION\_ERROR(-20000, 'Nu a fost găsit un angajat cu numele "'||nume\_angajator||'"'||'.');              WHEN 1 THEN                  manager\_gasit := rezultate(1);              ELSE                  DECLARE                      err\_msg VARCHAR2(255);                  BEGIN                      err\_msg := 'Multiplii angajați au numele "'||nume\_angajator||'":';                      FOR i IN 1..rezultate.COUNT                      LOOP                          err\_msg := err\_msg||' '||rezultate(i);                      END LOOP;                      RAISE\_APPLICATION\_ERROR(-20003, err\_msg);                  EXCEPTION                      WHEN VALUE\_ERROR THEN                          RAISE\_APPLICATION\_ERROR(-20003, err\_msg || '[...]');                  END;          END CASE;          SELECT COUNT(id\_angajat)          INTO este\_manager          FROM manager          WHERE id\_angajat = manager\_gasit;          IF este\_manager = 0 THEN              RAISE\_APPLICATION\_ERROR(-20001, 'Angajatul '||nume\_angajator||' (ID '||manager\_gasit||') nu este manager.');          END IF;      END;      BEGIN          /\*          Patru tabele: CASIER -> COMANDA -> COMANDA\_INCLUDE\_RETETA (asociativ) -> RETETA          \*/          WITH casieri AS (              SELECT id\_angajat, SUM(pret \* nr) profit              FROM angajat              JOIN comanda              ON angajat.id\_angajat = comanda.id\_casier              JOIN comanda\_include\_reteta              USING (id\_comanda)              JOIN reteta              USING (id\_reteta)              WHERE job\_cod = 'CASIER' AND id\_angajator = manager\_gasit              GROUP BY id\_angajat              ORDER BY profit          )          SELECT id\_angajat          INTO casier\_gasit          FROM casieri          WHERE rownum = 1;      EXCEPTION          WHEN NO\_DATA\_FOUND THEN              RAISE\_APPLICATION\_ERROR(-20002, 'Managerul '||nume\_angajator||' (ID '||manager\_gasit||') nu are casieri.');      END;      RETURN casier\_gasit;  END gaseste\_casierul\_cel\_mai\_profitabil;  /  DECLARE      /\*      Excepțiile nu pot fi declarate global, deoarece nu se poate stoca o excepție în bază da date.      Deci blocurile care vor să le prindă trebuie să creeze excepțiile și să le știe codurile.      Când va fi creată versiunea cu pachet a acestei cerințe, vor putea fi declarate în specificația pachetului.      \*/      nume\_nu\_exista EXCEPTION;      angajat\_nu\_e\_manager EXCEPTION;      manager\_nu\_are\_casieri EXCEPTION;      PRAGMA EXCEPTION\_INIT (nume\_nu\_exista, -20000);      PRAGMA EXCEPTION\_INIT (angajat\_nu\_e\_manager, -20001);      PRAGMA EXCEPTION\_INIT (manager\_nu\_are\_casieri, -20002);      /\*      Cursorul extrage automat date de test, independent de datele din tabel, fără a hardcoda nume.      Se vor returna how\_many\_of\_each (sau dacă nu se poate, cât mai mulți) angajați pentru categoriile:      - manageri cu angajați      - manageri fără angajați      - angajați non-manager      - nume non-existent      Va funționa chiar dacă seed-ul RNG pentru generarea datelor angajaților este schimbat.      Excepția de nume ambiguu va fi testată separat deoarece necesită inserarea de date în tabel.      \*/      CURSOR test\_data(how\_many\_of\_each NUMBER)      IS      WITH are\_angajat\_da\_nu AS (          /\*          Nu e suficienăm să verificăm dacă e autorizat, nu garantează că a angajat pe cineva încă.          \*/          SELECT id\_angajat, nume, (              SELECT DECODE(COUNT(a2.id\_angajat), 0, 0, 1)              FROM angajat a2              WHERE id\_angajator = a1.id\_angajat          ) AS tip\_test          FROM angajat a1          WHERE job\_cod = 'MANAGER'      )      SELECT \*      FROM (          SELECT are\_angajat\_da\_nu.\*, row\_number() over (partition by tip\_test order by id\_angajat) repeat\_nr          FROM are\_angajat\_da\_nu      )      WHERE repeat\_nr <= how\_many\_of\_each      UNION      SELECT id\_angajat, nume, 2, rownum      FROM angajat      WHERE job\_cod != 'MANAGER' AND rownum <= how\_many\_of\_each      UNION      SELECT -1, 'NONEXISTENT ' ||LEVEL||' (TEST)', 3, LEVEL FROM DUAL      CONNECT BY LEVEL <= how\_many\_of\_each;      test\_case\_nr BINARY\_INTEGER := 1;      TYPE labels IS TABLE OF VARCHAR2(255) INDEX BY BINARY\_INTEGER;      test\_names labels := labels();  BEGIN      test\_names(0) := 'Manager fără angajați';      test\_names(1) := 'Manager cu angajați';      test\_names(2) := 'Non-manager';      test\_names(3) := 'Nume non-existent';      FOR test\_row IN test\_data(3)      LOOP          dbms\_output.PUT('Test '||LPAD(test\_case\_nr, 2)||                          ' - '||test\_names(test\_row.tip\_test)||                          ' - '||test\_row.nume||                          ' - rezultat test: ');          DECLARE              casier angajat.id\_angajat%TYPE;          BEGIN              casier := gaseste\_casierul\_cel\_mai\_profitabil(test\_row.nume);              dbms\_output.PUT\_LINE('ID casier = '||casier);          EXCEPTION              WHEN angajat\_nu\_e\_manager THEN                  dbms\_output.PUT\_LINE('!! eroare angajat\_nu\_e\_manager !! - '||SQLERRM);              WHEN manager\_nu\_are\_casieri THEN                  dbms\_output.PUT\_LINE('!! eroare manager\_nu\_are\_casieri !! - '||SQLERRM);              WHEN nume\_nu\_exista THEN                  dbms\_output.PUT\_LINE('!! eroare nume\_nu\_exista !! - '||SQLERRM);          END;          test\_case\_nr := test\_case\_nr + 1;      END LOOP;  END;  /\*   Testare caz nume ambiguu. Se vor insera date ambigue, se va apela funcția, apoi se va face ROLLBACK.  \*/  INSERT INTO angajat (id\_angajat, id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu)  SELECT 1000 + LEVEL, 1, NULL, 'MANAGER', 'Test Ambiguu', SYSDATE, 0  FROM DUAL  CONNECT BY LEVEL <= 100;  INSERT INTO manager (id\_angajat, autorizat\_sa\_angajeze)  SELECT id\_angajat, 1 FROM angajat  WHERE nume = 'Test Ambiguu';  DECLARE      nume\_ambiguu EXCEPTION;      PRAGMA EXCEPTION\_INIT (nume\_ambiguu, -20003);  BEGIN      dbms\_output.PUT('Test ambiguu: ');      BEGIN          dbms\_output.PUT\_LINE('(nu ar trebui să se afișeze asta) ID casier = ' ||                               gaseste\_casierul\_cel\_mai\_profitabil('Test Ambiguu'));      EXCEPTION          WHEN nume\_ambiguu THEN              dbms\_output.PUT\_LINE('!! nume\_ambiguu !! - '||SQLERRM);      END;  END;  ROLLBACK;    DROP PROCEDURE gaseste\_casierul\_cel\_mai\_profitabil; |



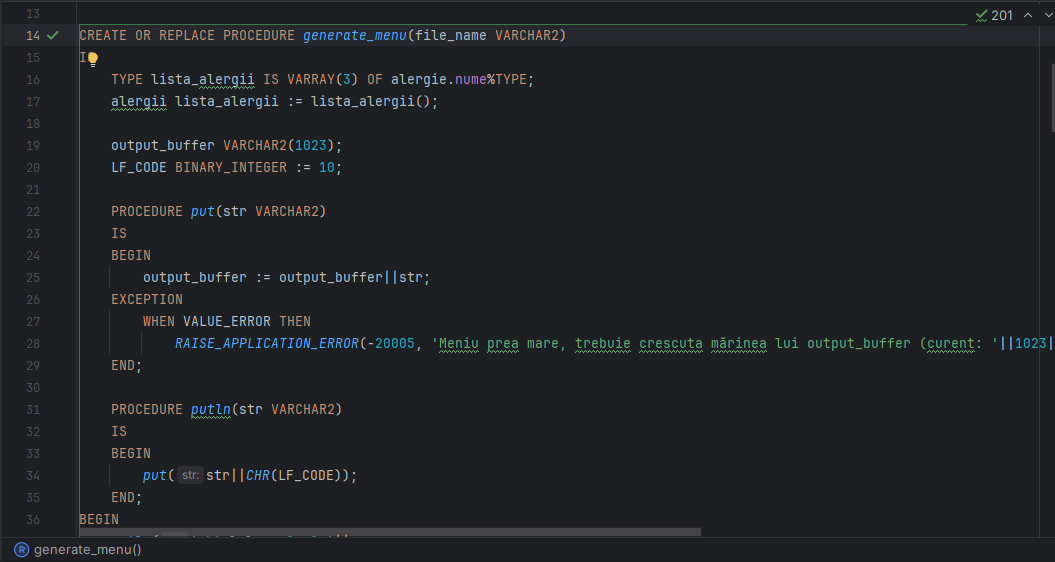
DataGrip crede incorect că nu se poate folosi INTO cu WITH.

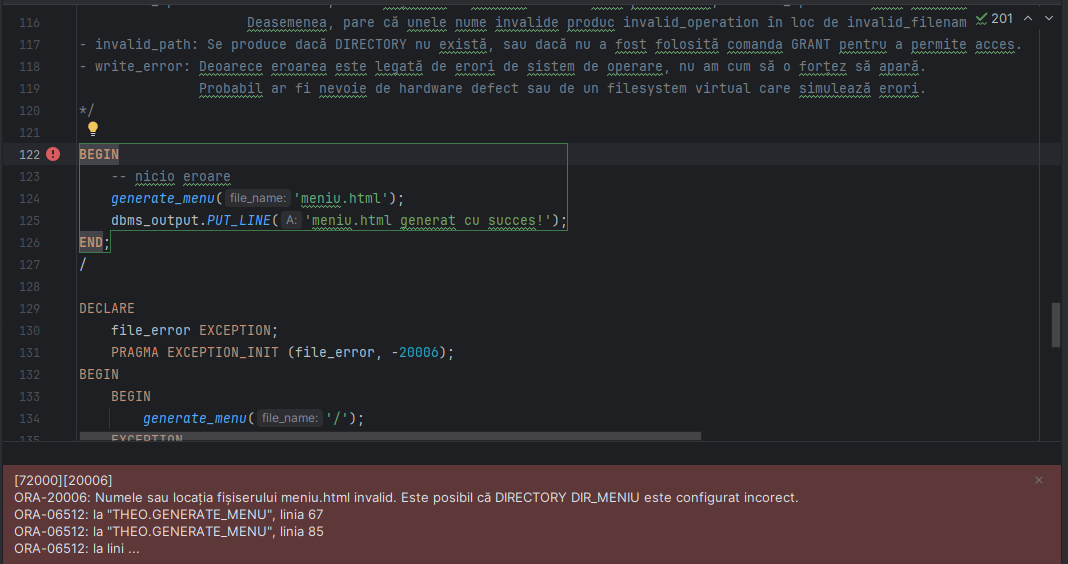


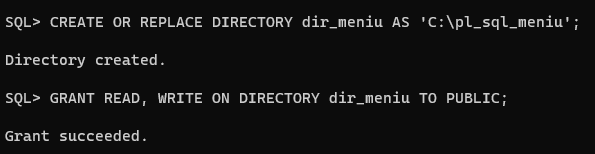
## Subprogram stocat – generare meniu

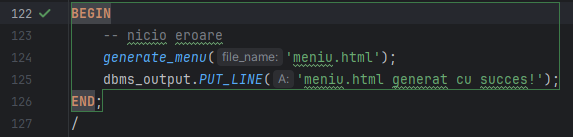
Să se genereze un meniu pentru restaurant. Trebuie listate rețetele, prețul lor, ingredientele și alergiile asociate fiecărei rețete. Se va genera în format HTML și se va exporta automat.

|  |
| --- |
| /\*  Să se genereze un meniu pentru restaurant.  Trebuie listate rețetele, prețul lor, ingredientele și alergiile asociate fiecărei rețete.  Se va genera în format HTML și se va exporta automat.  \*/  /\*  !!! Aceste comenzi necesita privilegii de administrator. Trebuie executate din SQL\*Plus folosind un cont de administrator. !!!  CREATE OR REPLACE DIRECTORY dir\_meniu AS 'C:\pl\_sql\_meniu';  GRANT READ, WRITE ON DIRECTORY dir\_meniu TO PUBLIC;  \*/  CREATE OR REPLACE PROCEDURE generate\_menu(file\_name VARCHAR2)  IS      TYPE lista\_alergii IS VARRAY(3) OF alergie.nume%TYPE;      alergii lista\_alergii := lista\_alergii();      output\_buffer VARCHAR2(1023);      LF\_CODE BINARY\_INTEGER := 10;      PROCEDURE put(str VARCHAR2)      IS      BEGIN          output\_buffer := output\_buffer||str;      EXCEPTION          WHEN VALUE\_ERROR THEN              RAISE\_APPLICATION\_ERROR(-20005, 'Meniu prea mare, trebuie crescuta mărinea lui output\_buffer (curent: '||1023||')');      END;      PROCEDURE putln(str VARCHAR2)      IS      BEGIN          put(str||CHR(LF\_CODE));      END;  BEGIN      putln('<html lang="ro">'||              '<head><meta charset="UTF-8"></head>'||              '<body>'||              '<h1>Bine ați venit la restaurantul nostru!</h1>'||              '<p>Oferim următoarele rețete:</p>'||              '<ul>'      );      FOR row\_reteta IN (SELECT id\_reteta, nume, pret FROM reteta)      LOOP          putln('<li><p>'||INITCAP(row\_reteta.nume)||' - '||row\_reteta.pret||' RON</p>');          /\*          Cinci tablele: RETETA -> RETETA\_CONTINE\_INGREDIENT -> INGREDIENT -> INGREDIENT\_PROVOACA\_ALERGIE -> ALERGIE          \*/          alergii.DELETE;          SELECT DISTINCT INITCAP(alergie.nume)          BULK COLLECT INTO alergii          FROM reteta          JOIN reteta\_contine\_ingredient          USING (id\_reteta)          JOIN ingredient          USING (id\_ingredient)          JOIN ingredient\_provoaca\_alergie          USING (id\_ingredient)          JOIN alergie          USING (id\_alergie)          WHERE id\_reteta = row\_reteta.id\_reteta;          IF alergii.count > 0          THEN              put('<p><em>Informații alergii: '||alergii(1));              FOR i IN 2..alergii.last              LOOP                  put(', '||alergii(i));              END LOOP;              putln('.</em></p>');          END IF;          putln('</li>');      END LOOP;      putln('</body></html>');      DECLARE          f UTL\_FILE.file\_type;          PROCEDURE rs(template\_str VARCHAR2) /\* prescurtare pentru raise \*/          IS          BEGIN              RAISE\_APPLICATION\_ERROR(-20006, REPLACE(template\_str, 'FNAME', file\_name));          END;      BEGIN          f := UTL\_FILE.fopen('DIR\_MENIU', file\_name, 'W');          UTL\_FILE.put(f, output\_buffer);          UTL\_FILE.FCLOSE(f);      EXCEPTION          WHEN UTL\_FILE.access\_denied THEN              rs('Serverul Oracle nu are access pentru a crea fișierul FNAME. Posibil locația DIRECTORY DIR\_MENIU trebuie schimbată.');          WHEN UTL\_FILE.file\_open THEN              rs('Fișierul FNAME este deja deschis. Nu se poate suprascrie.');          WHEN UTL\_FILE.internal\_error THEN              rs('Eroare necunoscută PL/SQL. Nu se poate genera fișierul FNAME.');          WHEN UTL\_FILE.invalid\_filename THEN              rs('Numele FNAME pentru fișier este invalid.');          WHEN UTL\_FILE.invalid\_operation THEN              rs('Operația pe fișierul FNAME nu a putut fi procesată din motive necunoscute. Verfică dacă folderul lui DIRECTORY DIR\_MENIU există, și permisiunile la folder.');          WHEN UTL\_FILE.invalid\_path THEN              rs('Numele sau locația fișiserului FNAME invalid. Este posibil că DIRECTORY DIR\_MENIU este configurat incorect.');          WHEN UTL\_FILE.write\_error THEN              rs('Nu s-a putut genera FNAME din cauza unei erori de sistem de operare la scrierea fișierului.');      END;  END;  /  /\*  Majoritatea erorilor UTL\_FILE nu pot fi testate într-un bloc:  - access\_denied: Nu pare că există pe sistemul Windows, ci eroarea invalid\_operation se ridică când fișierul nu poate fi modificat.                   E posibil că access\_denied există doar pe sisteme Linux.  - file\_open: Similar, nu pare că există. Când încerc să produc eroarea pe sistem Windows, obțin invalid\_operation.  - internal\_error: Deoarece această eroare este pentru erori nespecificate în implementarea lui UTL\_FILE,                    nu știu cum să o produc.  - invalid\_filename: Se produce ușor, însă pare că doar nume ce conțin / produc eroarea.                      Bănuiesc că UTL\_FILE presupune că Windows permite orice nume ce nu conține /, ca pe un sistem Linux.                      Alte nume invalide produc invalid\_operation, probabil deoarece UTL\_FILE întelege răspunsul de la Windows.  - invalid\_operation: Pe Windows, dacă fișierul e deschis sau nu avem permisiuni, invalid\_operation este ridicat.                       Deasemenea, pare că unele nume invalide produc invalid\_operation în loc de invalid\_filename.  - invalid\_path: Se produce dacă DIRECTORY nu există, sau dacă nu a fost folosită comanda GRANT pentru a permite acces.  - write\_error: Deoarece eroarea este legată de erori de sistem de operare, nu am cum să o forțez să apară.                 Probabil ar fi nevoie de hardware defect sau de un filesystem virtual care simulează erori.  \*/  BEGIN      -- nicio eroare      generate\_menu('meniu.html');      dbms\_output.PUT\_LINE('meniu.html generat cu succes!');      -- invalid\_filename  END;  /  DECLARE      file\_error EXCEPTION;      PRAGMA EXCEPTION\_INIT (file\_error, -20006);  BEGIN      BEGIN          generate\_menu('/');      EXCEPTION          WHEN file\_error THEN              dbms\_output.PUT\_LINE(SQLERRM);      END;      BEGIN          generate\_menu('?');      EXCEPTION          WHEN file\_error THEN              dbms\_output.PUT\_LINE(SQLERRM);      END;  END;  /\*  Eroarea needs\_increased\_output\_buffer se generează ușor prin umplerea tabelului RETETA cu date extra.  \*/  INSERT INTO reteta (id\_reteta, nume, pret)  SELECT 1000 + LEVEL, 'Test row '||LEVEL, 0  FROM DUAL  CONNECT BY LEVEL <= 1000;  DECLARE      needs\_increased\_output\_buffer EXCEPTION;      PRAGMA EXCEPTION\_INIT (needs\_increased\_output\_buffer, -20005);  BEGIN      generate\_menu('meniu.html');  EXCEPTION      WHEN needs\_increased\_output\_buffer THEN          dbms\_output.PUT\_LINE(SQLERRM);  END;  ROLLBACK;  DROP PROCEDURE generate\_menu; |

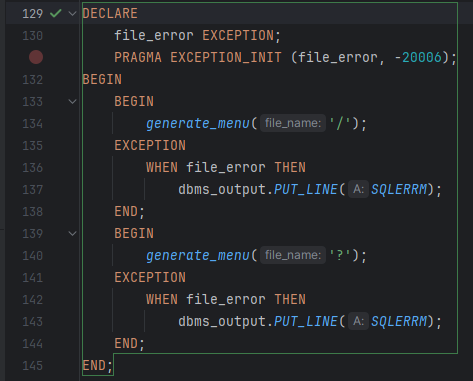




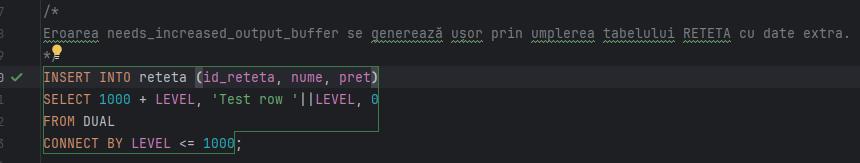


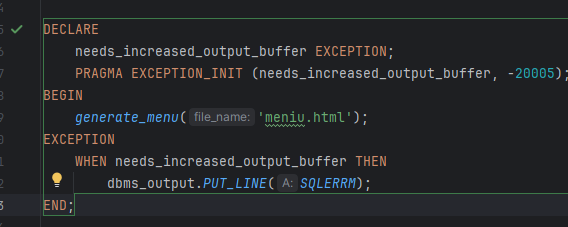














## Trigger – monitorizare modificări

S-a decis că deoarece unele tabele din baza de date se vor schimba rar, ar trebui monitorizate modificări ale lor.

Acele tabele sunt:

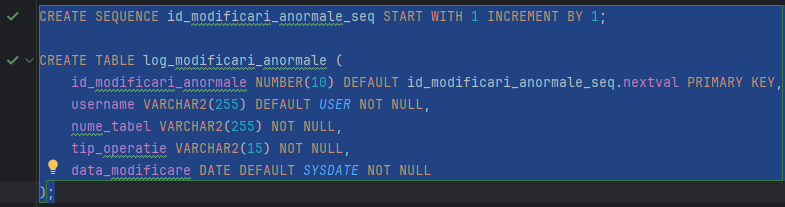
* ALERGIE
* INGREDIENT\_provoaca\_ALERGIE
* INGREDIENT
* RETETA\_contine\_INGREDIENT
* RETETA
* JOB

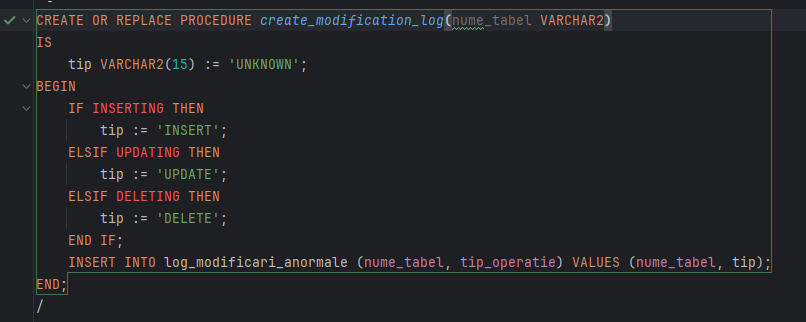
Să se stocheze log-ul tuturor modificărilor la aceste tabele.

Log-ul va include:

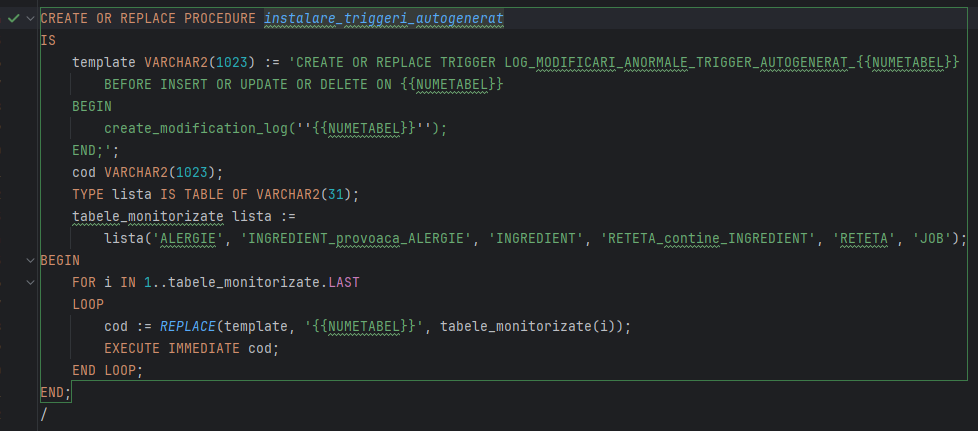
* username
* nume tabel
* tip operatie
* data modificare

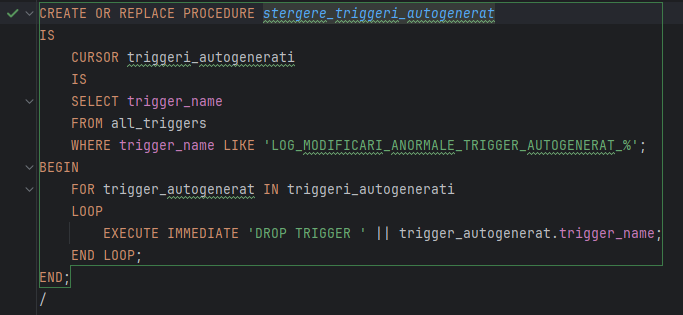
|  |
| --- |
| /\*  Să se genereze un meniu pentru restaurant.  Trebuie listate rețetele, prețul lor, ingredientele și alergiile asociate fiecărei rețete.  Se va genera în format HTML și se va exporta automat.  \*/  /\*  !!! Aceste comenzi necesita privilegii de administrator. Trebuie executate din SQL\*Plus folosind un cont de administrator. !!!  CREATE OR REPLACE DIRECTORY dir\_meniu AS 'C:\pl\_sql\_meniu';  GRANT READ, WRITE ON DIRECTORY dir\_meniu TO PUBLIC;  \*/  CREATE OR REPLACE PROCEDURE generate\_menu(file\_name VARCHAR2)  IS      TYPE lista\_alergii IS VARRAY(3) OF alergie.nume%TYPE;      alergii lista\_alergii := lista\_alergii();      output\_buffer VARCHAR2(1023);      LF\_CODE BINARY\_INTEGER := 10;      PROCEDURE put(str VARCHAR2)      IS      BEGIN          output\_buffer := output\_buffer||str;      EXCEPTION          WHEN VALUE\_ERROR THEN              RAISE\_APPLICATION\_ERROR(-20005, 'Meniu prea mare, trebuie crescuta mărinea lui output\_buffer (curent: '||1023||')');      END;      PROCEDURE putln(str VARCHAR2)      IS      BEGIN          put(str||CHR(LF\_CODE));      END;  BEGIN      putln('<html lang="ro">'||              '<head><meta charset="UTF-8"></head>'||              '<body>'||              '<h1>Bine ați venit la restaurantul nostru!</h1>'||              '<p>Oferim următoarele rețete:</p>'||              '<ul>'      );      FOR row\_reteta IN (SELECT id\_reteta, nume, pret FROM reteta)      LOOP          putln('<li><p>'||INITCAP(row\_reteta.nume)||' - '||row\_reteta.pret||' RON</p>');          /\*          Cinci tablele: RETETA -> RETETA\_CONTINE\_INGREDIENT -> INGREDIENT -> INGREDIENT\_PROVOACA\_ALERGIE -> ALERGIE          \*/          alergii.DELETE;          SELECT DISTINCT INITCAP(alergie.nume)          BULK COLLECT INTO alergii          FROM reteta          JOIN reteta\_contine\_ingredient          USING (id\_reteta)          JOIN ingredient          USING (id\_ingredient)          JOIN ingredient\_provoaca\_alergie          USING (id\_ingredient)          JOIN alergie          USING (id\_alergie)          WHERE id\_reteta = row\_reteta.id\_reteta;          IF alergii.count > 0          THEN              put('<p><em>Informații alergii: '||alergii(1));              FOR i IN 2..alergii.last              LOOP                  put(', '||alergii(i));              END LOOP;              putln('.</em></p>');          END IF;          putln('</li>');      END LOOP;      putln('</body></html>');      DECLARE          f UTL\_FILE.file\_type;          PROCEDURE rs(template\_str VARCHAR2) /\* prescurtare pentru raise \*/          IS          BEGIN              RAISE\_APPLICATION\_ERROR(-20006, REPLACE(template\_str, 'FNAME', file\_name));          END;      BEGIN          f := UTL\_FILE.fopen('DIR\_MENIU', file\_name, 'W');          UTL\_FILE.put(f, output\_buffer);          UTL\_FILE.FCLOSE(f);      EXCEPTION          WHEN UTL\_FILE.access\_denied THEN              rs('Serverul Oracle nu are access pentru a crea fișierul FNAME. Posibil locația DIRECTORY DIR\_MENIU trebuie schimbată.');          WHEN UTL\_FILE.file\_open THEN              rs('Fișierul FNAME este deja deschis. Nu se poate suprascrie.');          WHEN UTL\_FILE.internal\_error THEN              rs('Eroare necunoscută PL/SQL. Nu se poate genera fișierul FNAME.');          WHEN UTL\_FILE.invalid\_filename THEN              rs('Numele FNAME pentru fișier este invalid.');          WHEN UTL\_FILE.invalid\_operation THEN              rs('Operația pe fișierul FNAME nu a putut fi procesată din motive necunoscute. Verfică dacă folderul lui DIRECTORY DIR\_MENIU există, și permisiunile la folder.');          WHEN UTL\_FILE.invalid\_path THEN              rs('Numele sau locația fișiserului FNAME invalid. Este posibil că DIRECTORY DIR\_MENIU este configurat incorect.');          WHEN UTL\_FILE.write\_error THEN              rs('Nu s-a putut genera FNAME din cauza unei erori de sistem de operare la scrierea fișierului.');      END;  END;  /  /\*  Majoritatea erorilor UTL\_FILE nu pot fi testate într-un bloc:  - access\_denied: Nu pare că există pe sistemul Windows, ci eroarea invalid\_operation se ridică când fișierul nu poate fi modificat.                   E posibil că access\_denied există doar pe sisteme Linux.  - file\_open: Similar, nu pare că există. Când încerc să produc eroarea pe sistem Windows, obțin invalid\_operation.  - internal\_error: Deoarece această eroare este pentru erori nespecificate în implementarea lui UTL\_FILE,                    nu știu cum să o produc.  - invalid\_filename: Se produce ușor, însă pare că doar nume ce conțin / produc eroarea.                      Bănuiesc că UTL\_FILE presupune că Windows permite orice nume ce nu conține /, ca pe un sistem Linux.                      Alte nume invalide produc invalid\_operation, probabil deoarece UTL\_FILE întelege răspunsul de la Windows.  - invalid\_operation: Pe Windows, dacă fișierul e deschis sau nu avem permisiuni, invalid\_operation este ridicat.                       Deasemenea, pare că unele nume invalide produc invalid\_operation în loc de invalid\_filename.  - invalid\_path: Se produce dacă DIRECTORY nu există, sau dacă nu a fost folosită comanda GRANT pentru a permite acces.  - write\_error: Deoarece eroarea este legată de erori de sistem de operare, nu am cum să o forțez să apară.                 Probabil ar fi nevoie de hardware defect sau de un filesystem virtual care simulează erori.  \*/  BEGIN      -- nicio eroare      generate\_menu('meniu.html');      dbms\_output.PUT\_LINE('meniu.html generat cu succes!');  END;  /  DECLARE      file\_error EXCEPTION;      PRAGMA EXCEPTION\_INIT (file\_error, -20006);  BEGIN      BEGIN          generate\_menu('/');      EXCEPTION          WHEN file\_error THEN              dbms\_output.PUT\_LINE(SQLERRM);      END;      BEGIN          generate\_menu('?');      EXCEPTION          WHEN file\_error THEN              dbms\_output.PUT\_LINE(SQLERRM);      END;  END;  /\*  Eroarea needs\_increased\_output\_buffer se generează ușor prin umplerea tabelului RETETA cu date extra.  \*/  INSERT INTO reteta (id\_reteta, nume, pret)  SELECT 1000 + LEVEL, 'Test row '||LEVEL, 0  FROM DUAL  CONNECT BY LEVEL <= 1000;  DECLARE      needs\_increased\_output\_buffer EXCEPTION;      PRAGMA EXCEPTION\_INIT (needs\_increased\_output\_buffer, -20005);  BEGIN      generate\_menu('meniu.html');  EXCEPTION      WHEN needs\_increased\_output\_buffer THEN          dbms\_output.PUT\_LINE(SQLERRM);  END;  ROLLBACK;  DROP PROCEDURE generate\_menu; |

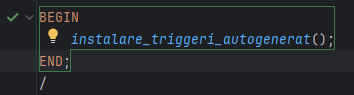


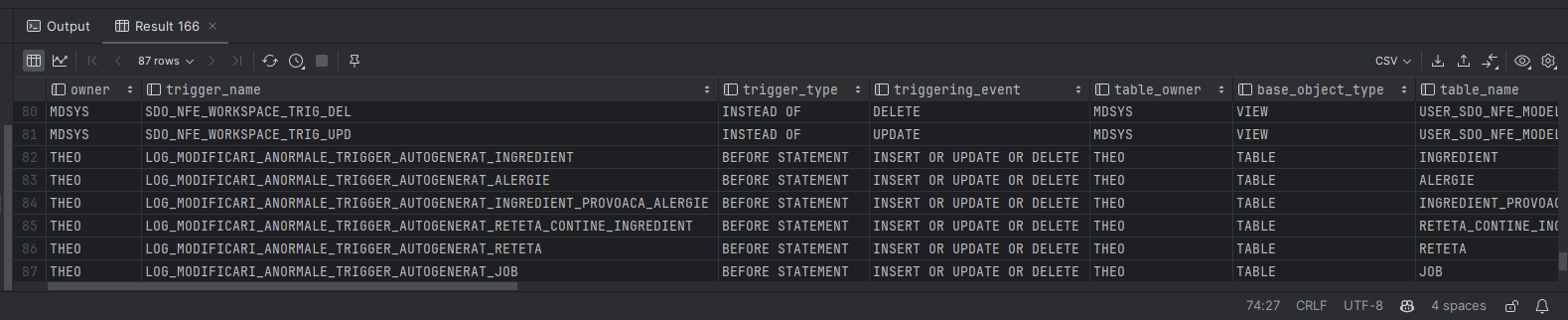


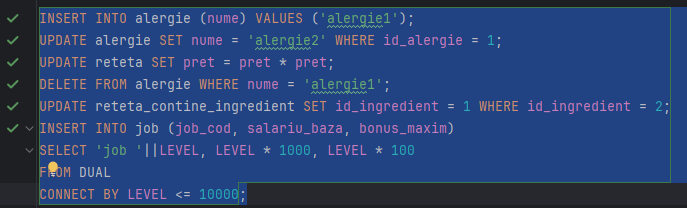
DataGrip nu recunoaște INSERTING, UPDATING, DELETING și le marchează incorect ca erori.

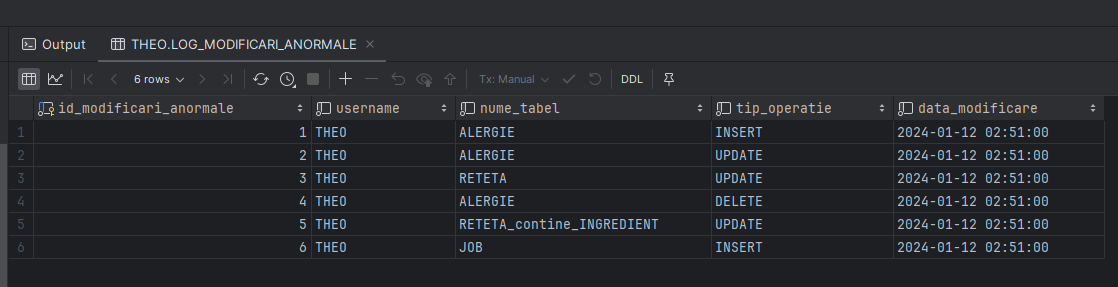




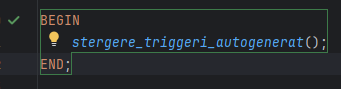


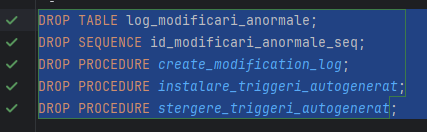












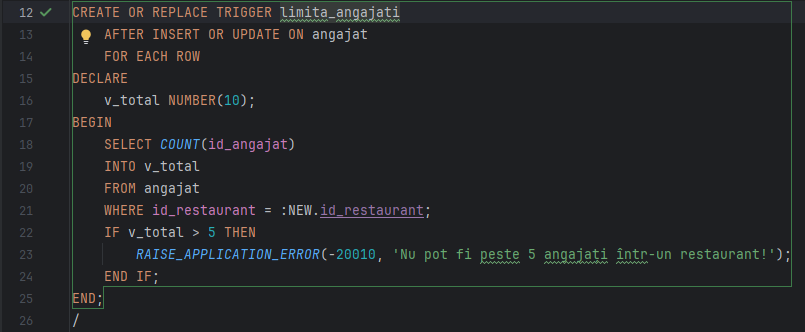
## Trigger – max angajați

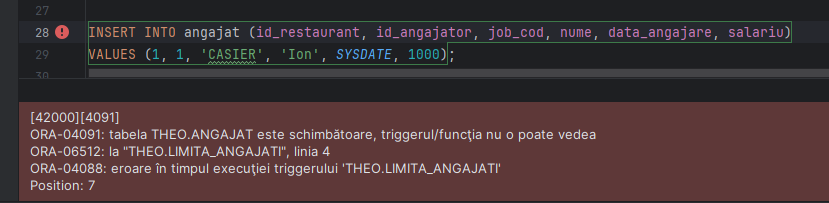
Din cauza unor griji că managerii nu sunt antrenați să organizeze numere mari de angajați, s-a decis să se limiteze numărul de angajați la 5 pe restaurant.

Să se creeze un trigger care să nu permită peste 5 angajați pe restaurant.

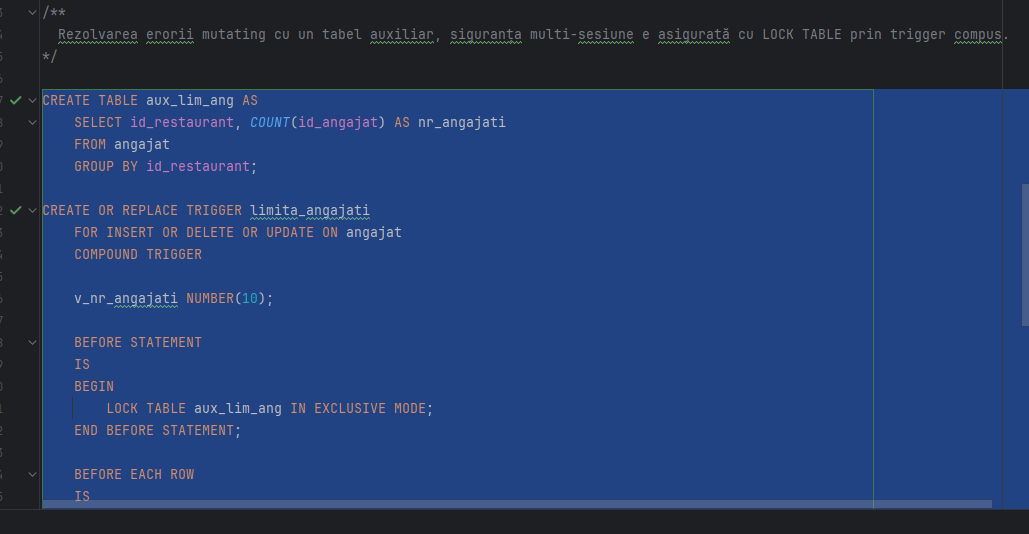
Se va arăta și un exemplu de trigger \_incorect\_, care nu funcționează din cauze erorii mutating.

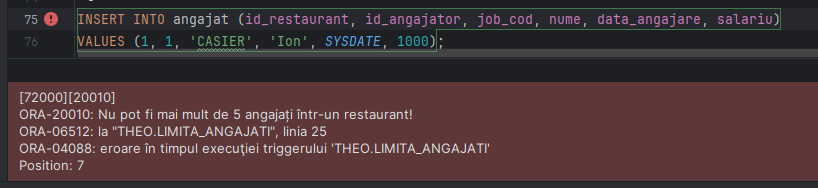
|  |
| --- |
| /\*  Din cauza unor griji că managerii nu sunt antrenați să organizeze numere mari de angajați,  s-a decis să se limiteze numărul de angajați la 5 pe restaurant.  Să se creeze un trigger care să nu permită peste 5 angajați pe restaurant.  Se va arăta și un exemplu de trigger \_incorect\_, care nu funcționează din cauze erorii mutating.  \*/  /\*\*    Exemplu de trigger cu eroare mutating.  \*/  CREATE OR REPLACE TRIGGER limita\_angajati      AFTER INSERT OR UPDATE ON angajat      FOR EACH ROW  DECLARE      v\_total NUMBER(10);  BEGIN      SELECT COUNT(id\_angajat)      INTO v\_total      FROM angajat      WHERE id\_restaurant = :NEW.id\_restaurant;      IF v\_total > 5 THEN          RAISE\_APPLICATION\_ERROR(-20010, 'Nu pot fi peste 5 angajați într-un restaurant!');      END IF;  END;  /  INSERT INTO angajat (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu)  VALUES (1, 1, 'CASIER', 'Ion', SYSDATE, 1000);  DROP TRIGGER limita\_angajati;  /\*\*    Rezolvarea erorii mutating cu un tabel auxiliar, siguranța multi-sesiune e asigurată cu LOCK TABLE prin trigger compus.  \*/  CREATE TABLE aux\_lim\_ang AS      SELECT id\_restaurant, COUNT(id\_angajat) AS nr\_angajati      FROM angajat      GROUP BY id\_restaurant;  CREATE OR REPLACE TRIGGER limita\_angajati      FOR INSERT OR DELETE OR UPDATE ON angajat      COMPOUND TRIGGER      v\_nr\_angajati NUMBER(10);      BEFORE STATEMENT      IS      BEGIN          LOCK TABLE aux\_lim\_ang IN EXCLUSIVE MODE;      END BEFORE STATEMENT;      BEFORE EACH ROW      IS      BEGIN          IF DELETING OR UPDATING THEN              UPDATE aux\_lim\_ang              SET nr\_angajati = nr\_angajati - 1              WHERE id\_restaurant = :OLD.id\_restaurant;          END IF;          IF INSERTING OR UPDATING THEN              UPDATE aux\_lim\_ang              SET nr\_angajati = nr\_angajati + 1              WHERE id\_restaurant = :NEW.id\_restaurant              RETURNING nr\_angajati INTO v\_nr\_angajati;              IF v\_nr\_angajati > 5 THEN                  raise\_application\_error(-20010, 'Nu pot fi mai mult de 5 angajați într-un restaurant!');              END IF;          END IF;      END BEFORE EACH ROW;  END;  /  INSERT INTO angajat (id\_restaurant, id\_angajator, job\_cod, nume, data\_angajare, salariu)  VALUES (1, 1, 'CASIER', 'Ion', SYSDATE, 1000);  DROP TRIGGER limita\_angajati;  DROP TABLE aux\_lim\_ang; |













## Trigger – protejarea tabelelor

Să se interzică ștergerea și alterarea tabelelor din baza de date.

Încercările de ștergere sau alterare a tabelelor vor fi înregistrate într-o tabelă log.

Log-ul trebuie protejat la fel ca și restul tabelelor.

|  |
| --- |
| /\*  Să se interzică ștergerea și alterarea tabelelor din baza de date.  Încercările de ștergere sau alterare a tabelelor vor fi înregistrate într-o tabelă log.  Log-ul trebuie protejat la fel ca și restul tabelelor.  \*/  CREATE SEQUENCE incercari\_log\_seq START WITH 1 INCREMENT BY 1;  CREATE TABLE incercari\_log (      id\_incercare NUMBER(10) DEFAULT incercari\_log\_seq.nextval PRIMARY KEY,      username VARCHAR2(127) DEFAULT user NOT NULL,      data date DEFAULT sysdate NOT NULL,      operation VARCHAR2(127) NOT NULL,      obj\_type VARCHAR2(127) NOT NULL,      obj\_name VARCHAR2(127) NOT NULL  );  CREATE OR REPLACE PROCEDURE log\_ddl\_neanulabil  IS      PRAGMA AUTONOMOUS\_TRANSACTION;  BEGIN      INSERT INTO incercari\_log (operation, obj\_type, obj\_name)      VALUES (sys.sysevent, sys.DICTIONARY\_OBJ\_TYPE(), sys.DICTIONARY\_OBJ\_NAME());      COMMIT;  END;  /  CREATE OR REPLACE TRIGGER protectie\_tabele      BEFORE DDL ON THEO.SCHEMA  DECLARE      TYPE lista\_protejate IS TABLE OF VARCHAR2(127);      lista lista\_protejate := lista\_protejate('ALERGIE', 'ANGAJAT', 'BUCATAR', 'CASIER', 'COMANDA',          'COMANDA\_INCLUDE\_RETETA', 'INGREDIENT', 'INGREDIENT\_PROVOACA\_ALERGIE', 'JOB', 'LIVRARE',          'MANAGER', 'ORAS', 'RESTAURANT', 'RETETA', 'RETETA\_CONTINE\_INGREDIENT', 'INCERCARI\_LOG');  BEGIN      FOR i IN 1..lista.COUNT      LOOP          IF (UPPER(sys.DICTIONARY\_OBJ\_NAME()) = lista(i))          THEN              log\_ddl\_neanulabil;              RAISE\_APPLICATION\_ERROR(-20011, 'Nu se poate folosi DDL pe tabelele protejate!');          END IF;      END LOOP;  END;  /  ALTER TABLE alergie ADD CONSTRAINT tmp CHECK (nume != '');  CREATE TABLE test (id NUMBER(10) PRIMARY KEY, nume VARCHAR2(127));  DROP TABLE test;  DROP TABLE incercari\_log;  SELECT \* FROM incercari\_log;  DROP TRIGGER protectie\_tabele;  DROP PROCEDURE log\_ddl\_neanulabil;  DROP TABLE incercari\_log;  DROP SEQUENCE incercari\_log\_seq; |



