**Proiect**

**Sisteme de Gestiune a Bazelor de Date (SGBD)**

**Gestionarea sejururilor din cadrul unui lant de hoteluri**

**Capatina Razvan-Nicolae**

**Grupa 252**

Cuprins

[Prezentarea bazei de date si precizarea utilitatii sale 4](#_Toc155889378)

[Diagrama Entitate-Relatie (ERD) 5](#_Toc155889379)

[Diagrama Conceptuala 6](#_Toc155889380)

[Implementarea diagramei conceptuale in Oracle 7](#_Toc155889381)

[Adaugarea informatiilor coerente in tabelele create 22](#_Toc155889382)

[Exercitiul 6 57](#_Toc155889383)

[Exercitiul 7 61](#_Toc155889384)

[Exercitiul 8 65](#_Toc155889385)

[Exercitiul 9 75](#_Toc155889386)

[Exercitiul 10 83](#_Toc155889387)

[Exercitiul 11 86](#_Toc155889388)

[Exercitiul 12 91](#_Toc155889389)

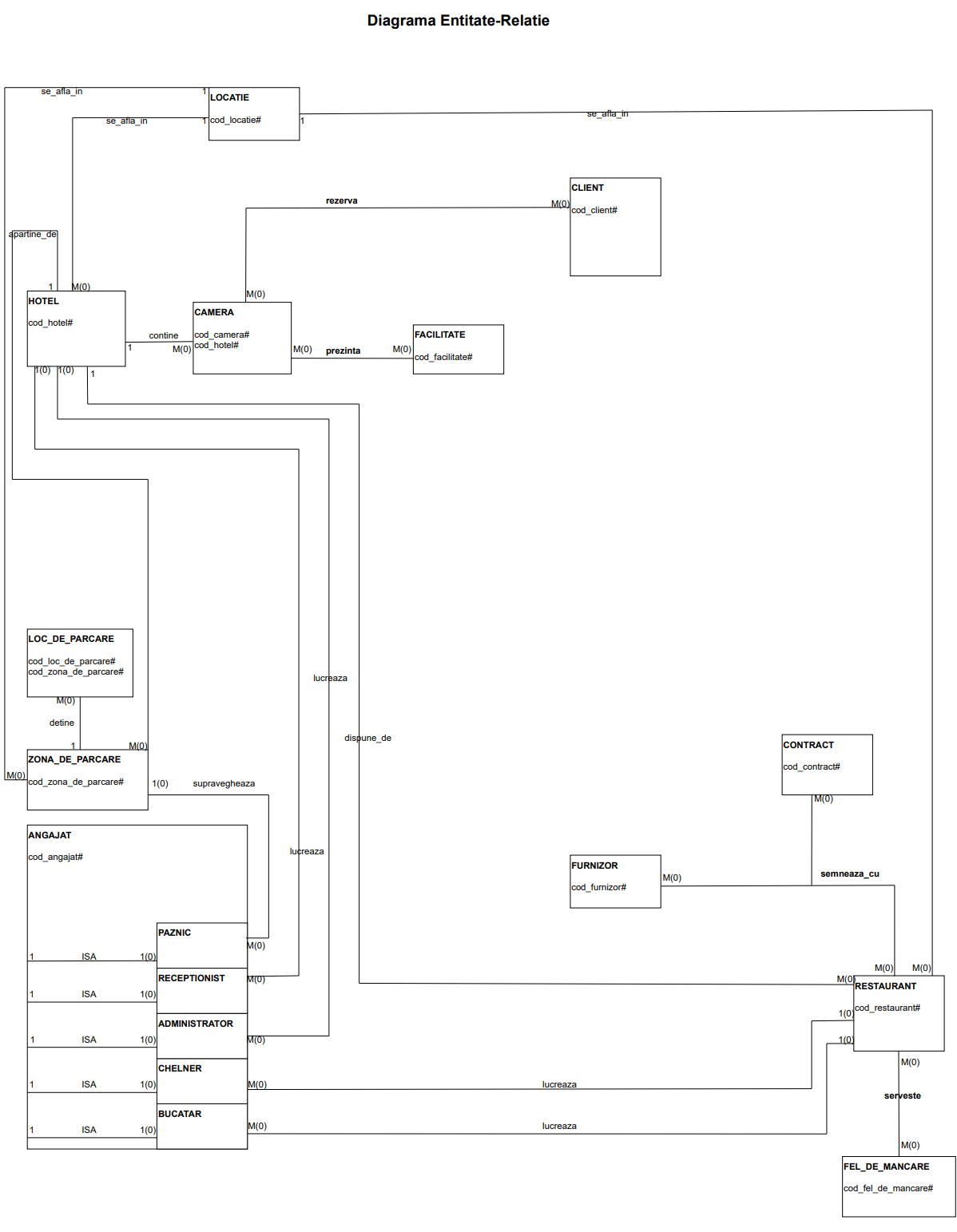
[Exercitiul 13 96](#_Toc155889390)

[Exercitiul 14 115](#_Toc155889391)

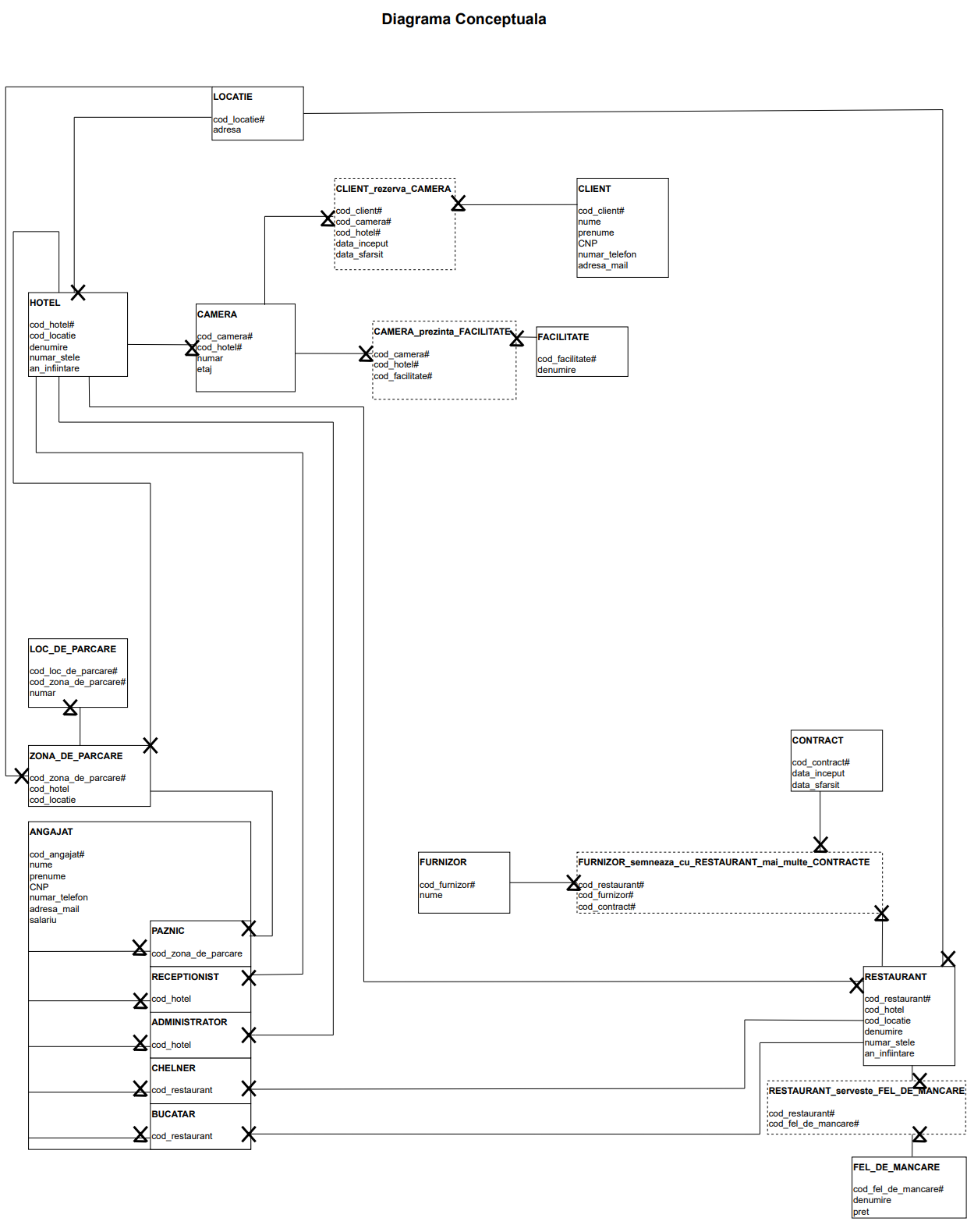
# Prezentarea bazei de date si precizarea utilitatii sale

Modelul de date va gestiona informatii legate de un lant de hoteluri. Vor fi luate in calcul ce hoteluri exista, ce camere apartin de fiecare hotel si daca au sau nu anumite facilitati. Fiecare hotel se va afla intr-o zona si va avea sau nu un restaurant si o zona de parcare formata din mai multe locuri. In cazul in care exista, restaurantul va servi mai multe feluri de mancare. De asemenea, vor exista mai multi angajati in cadrul lantului de hoteluri, acestia ocupand diverse functii. In cadrul restaurantului vor lucra bucatari si chelneri, in timp ce in cadrul hotelului vor lucra receptionisti si administratori. Zona de parcare, in cazul in care exista, va fi supravegheata de un paznic. O camera va putea fi rezervata de mai multe persoane. O persoana poate rezerva mai multe camere. Fiecare hotel, restaurant si zona de parcare va fi situata in exact o locatie. Furnizorii, in cazul existentei lor, vor putea incheia mai multe contracte cu mai multe restaurante din cadrul lantului de hoteluri, contribuind la aprovizionarea acestora. Un tip de contract va putea fi incheiat intre mai multi furnizori si mai multe restaurante, iar un restaurant poate realiza mai multe intelegeri cu diversi furnizori.

# Diagrama Entitate-Relatie (ERD)

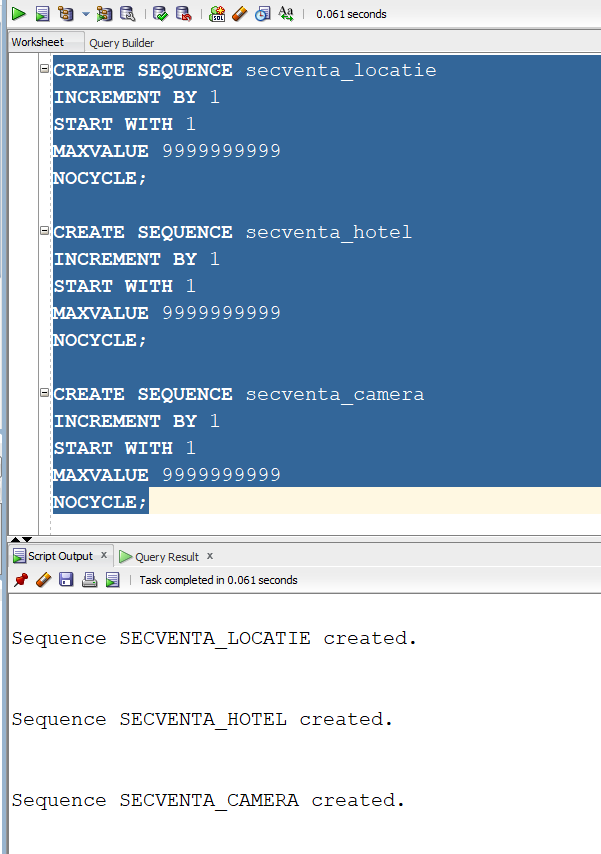


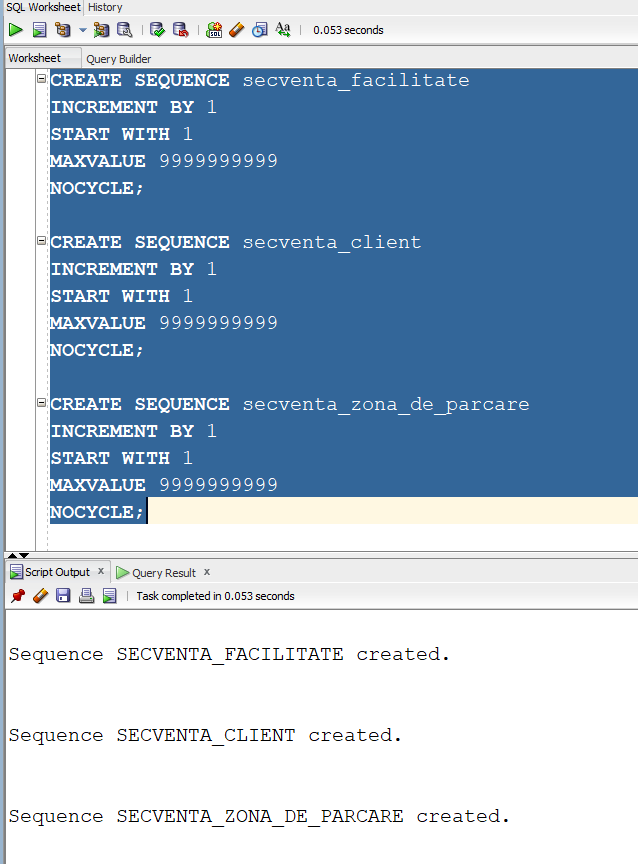
# Diagrama Conceptuala

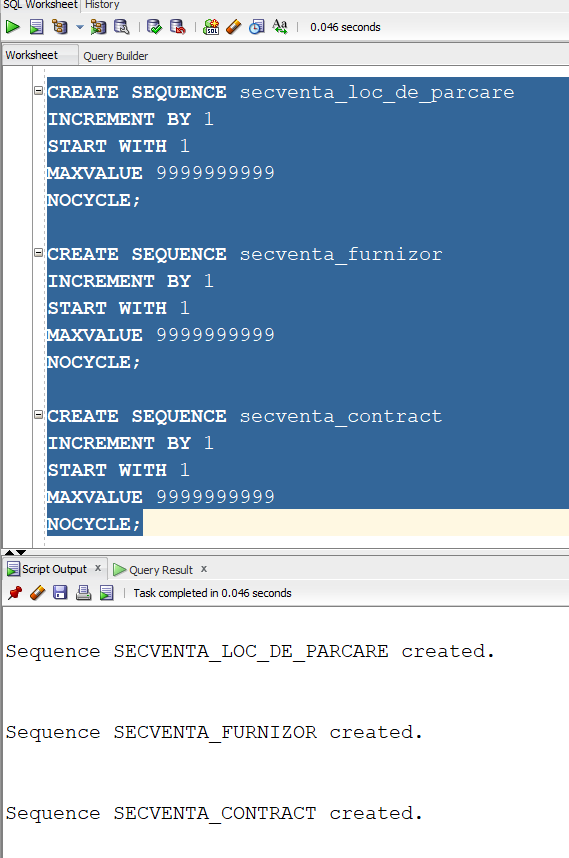


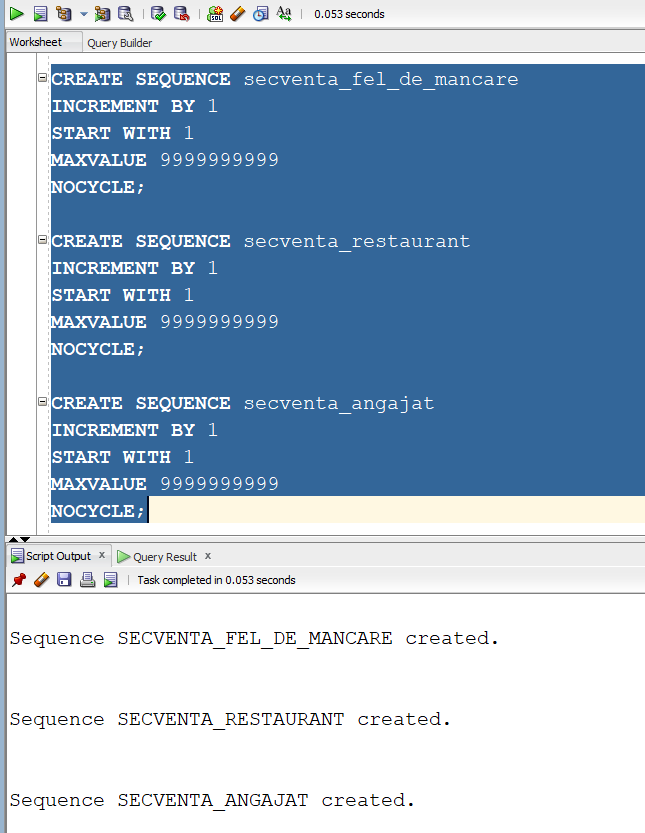
# Implementarea diagramei conceptuale in Oracle

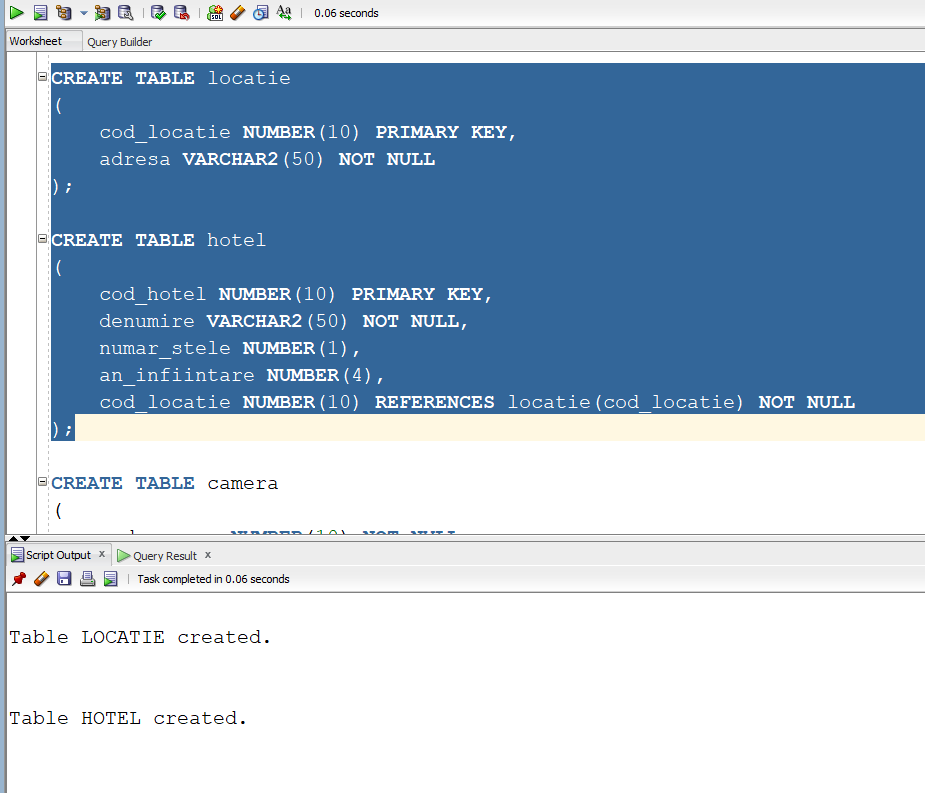
|  |
| --- |
| CREATE SEQUENCE secventa\_locatie  INCREMENT BY 1  START WITH 1  MAXVALUE 9999999999  NOCYCLE;    CREATE SEQUENCE secventa\_hotel  INCREMENT BY 1  START WITH 1  MAXVALUE 9999999999  NOCYCLE;    CREATE SEQUENCE secventa\_camera  INCREMENT BY 1  START WITH 1  MAXVALUE 9999999999  NOCYCLE;    CREATE SEQUENCE secventa\_facilitate  INCREMENT BY 1  START WITH 1  MAXVALUE 9999999999  NOCYCLE;    CREATE SEQUENCE secventa\_client  INCREMENT BY 1  START WITH 1  MAXVALUE 9999999999  NOCYCLE;    CREATE SEQUENCE secventa\_zona\_de\_parcare  INCREMENT BY 1  START WITH 1  MAXVALUE 9999999999  NOCYCLE;    CREATE SEQUENCE secventa\_loc\_de\_parcare  INCREMENT BY 1  START WITH 1  MAXVALUE 9999999999  NOCYCLE;    CREATE SEQUENCE secventa\_furnizor  INCREMENT BY 1  START WITH 1  MAXVALUE 9999999999  NOCYCLE;    CREATE SEQUENCE secventa\_contract  INCREMENT BY 1  START WITH 1  MAXVALUE 9999999999  NOCYCLE;    CREATE SEQUENCE secventa\_fel\_de\_mancare  INCREMENT BY 1  START WITH 1  MAXVALUE 9999999999  NOCYCLE;    CREATE SEQUENCE secventa\_restaurant  INCREMENT BY 1  START WITH 1  MAXVALUE 9999999999  NOCYCLE;    CREATE SEQUENCE secventa\_angajat  INCREMENT BY 1  START WITH 1  MAXVALUE 9999999999  NOCYCLE;    CREATE TABLE locatie  (  cod\_locatie NUMBER(10) PRIMARY KEY,  adresa VARCHAR2(50) NOT NULL  );    CREATE TABLE hotel  (  cod\_hotel NUMBER(10) PRIMARY KEY,  denumire VARCHAR2(50) NOT NULL,  numar\_stele NUMBER(1),  an\_infiintare NUMBER(4),  cod\_locatie NUMBER(10) REFERENCES locatie(cod\_locatie) NOT NULL  );    CREATE TABLE camera  (  cod\_camera NUMBER(10) NOT NULL,  cod\_hotel NUMBER(10) REFERENCES hotel(cod\_hotel) NOT NULL,  PRIMARY KEY (cod\_camera, cod\_hotel),  numar NUMBER(10) NOT NULL CHECK(numar > 0),  etaj NUMBER(3) DEFAULT 0 CHECK(etaj >= 0) NOT NULL  );    CREATE TABLE facilitate  (  cod\_facilitate NUMBER(10) PRIMARY KEY,  denumire VARCHAR2(50) NOT NULL  );    CREATE TABLE client  (  cod\_client NUMBER(10) PRIMARY KEY,  nume VARCHAR2(50) NOT NULL,  prenume VARCHAR2(50) NOT NULL,  cnp VARCHAR2(13),  numar\_telefon VARCHAR2(50) NOT NULL,  adresa\_mail VARCHAR2(50)  );    CREATE TABLE camera\_prezinta\_facilitate  (  cod\_camera NUMBER(10) NOT NULL,  cod\_hotel NUMBER(10) NOT NULL,  cod\_facilitate NUMBER(10) REFERENCES facilitate(cod\_facilitate) NOT NULL,  FOREIGN KEY(cod\_camera, cod\_hotel) REFERENCES camera(cod\_camera, cod\_hotel),  PRIMARY KEY(cod\_camera, cod\_hotel, cod\_facilitate)  );    CREATE TABLE client\_rezerva\_camera  (  cod\_client NUMBER(10) REFERENCES client(cod\_client) NOT NULL,  cod\_camera NUMBER(10) NOT NULL,  cod\_hotel NUMBER(10) NOT NULL,  data\_inceput DATE DEFAULT SYSDATE NOT NULL,  data\_sfarsit DATE NOT NULL,  FOREIGN KEY (cod\_camera, cod\_hotel) REFERENCES camera(cod\_camera, cod\_hotel),  CONSTRAINT check\_date\_client\_rezerva\_camera CHECK(data\_inceput <= data\_sfarsit),  PRIMARY KEY (cod\_client, cod\_camera, cod\_hotel, data\_inceput, data\_sfarsit)  );    CREATE TABLE zona\_de\_parcare  (  cod\_zona\_de\_parcare NUMBER(10) PRIMARY KEY,  cod\_hotel NUMBER(10) REFERENCES hotel(cod\_hotel) NOT NULL,  cod\_locatie NUMBER(10) REFERENCES locatie(cod\_locatie) NOT NULL  );    CREATE TABLE loc\_de\_parcare  (  cod\_loc\_de\_parcare NUMBER(10) NOT NULL,  cod\_zona\_de\_parcare NUMBER(10) NOT NULL,  PRIMARY KEY (cod\_loc\_de\_parcare, cod\_zona\_de\_parcare),  numar NUMBER(10) CHECK(numar > 0) NOT NULL  );    CREATE TABLE furnizor  (  cod\_furnizor NUMBER(10) PRIMARY KEY,  nume VARCHAR2(50) NOT NULL  );    CREATE TABLE contract  (  cod\_contract NUMBER(10) PRIMARY KEY,  data\_inceput DATE DEFAULT SYSDATE NOT NULL,  data\_sfarsit DATE NOT NULL,  CONSTRAINT check\_date\_contract CHECK(data\_inceput <= data\_sfarsit)  );    CREATE TABLE fel\_de\_mancare  (  cod\_fel\_de\_mancare NUMBER(10) PRIMARY KEY,  denumire VARCHAR2(50) NOT NULL,  pret NUMBER(5) NOT NULL  );    CREATE TABLE restaurant  (  cod\_restaurant NUMBER(10) PRIMARY KEY,  cod\_hotel NUMBER(10) REFERENCES hotel(cod\_hotel) NOT NULL,  cod\_locatie NUMBER(10) REFERENCES locatie(cod\_locatie) NOT NULL,  denumire VARCHAR2(50) NOT NULL,  numar\_stele NUMBER(1),  an\_infiintare NUMBER(4)  );    CREATE TABLE restaurant\_serveste\_fel\_de\_mancare  (  cod\_restaurant NUMBER(10) REFERENCES restaurant(cod\_restaurant) NOT NULL,  cod\_fel\_de\_mancare NUMBER(10) REFERENCES fel\_de\_mancare(cod\_fel\_de\_mancare) NOT NULL,  PRIMARY KEY (cod\_restaurant, cod\_fel\_de\_mancare)  );    CREATE TABLE furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte  (  cod\_restaurant NUMBER(10) REFERENCES restaurant(cod\_restaurant) NOT NULL,  cod\_furnizor NUMBER(10) REFERENCES furnizor(cod\_furnizor) NOT NULL,  cod\_contract NUMBER(10) REFERENCES contract(cod\_contract) NOT NULL,  PRIMARY KEY (cod\_restaurant, cod\_furnizor, cod\_contract)  );    CREATE TABLE angajat  (  cod\_angajat NUMBER(10) PRIMARY KEY,  nume VARCHAR2(50) NOT NULL,  prenume VARCHAR2(50) NOT NULL,  cnp VARCHAR2(13) NOT NULL,  numar\_telefon VARCHAR2(50) NOT NULL,  adresa\_mail VARCHAR2(50),  salariu NUMBER(10) NOT NULL  );    CREATE TABLE paznic  (  cod\_angajat NUMBER(10) PRIMARY KEY REFERENCES angajat(cod\_angajat),  cod\_zona\_de\_parcare NUMBER(10) REFERENCES zona\_de\_parcare(cod\_zona\_de\_parcare)  );    CREATE TABLE receptionist  (  cod\_angajat NUMBER(10) PRIMARY KEY REFERENCES angajat(cod\_angajat),  cod\_hotel NUMBER(10) REFERENCES hotel(cod\_hotel)  );    CREATE TABLE administrator  (  cod\_angajat NUMBER(10) PRIMARY KEY REFERENCES angajat(cod\_angajat),  cod\_hotel NUMBER(10) REFERENCES hotel(cod\_hotel)  );    CREATE TABLE chelner  (  cod\_angajat NUMBER(10) PRIMARY KEY REFERENCES angajat(cod\_angajat),  cod\_restaurant NUMBER(10) REFERENCES restaurant(cod\_restaurant)  );    CREATE TABLE bucatar  (  cod\_angajat NUMBER(10) PRIMARY KEY REFERENCES angajat(cod\_angajat),  cod\_restaurant NUMBER(10) REFERENCES restaurant(cod\_restaurant)  ); |

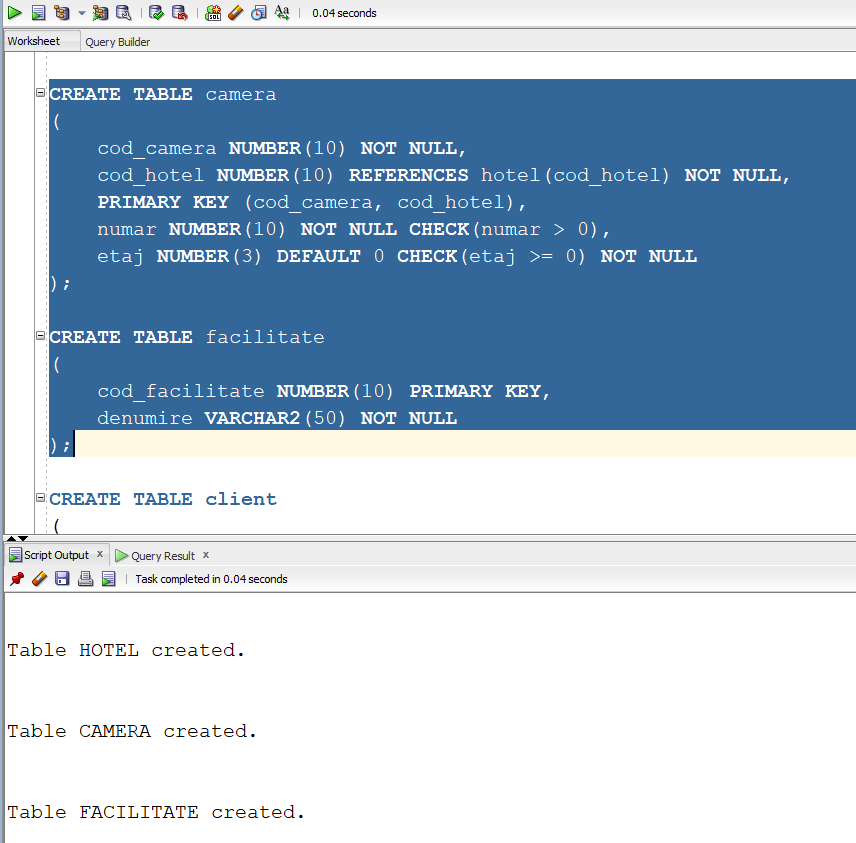


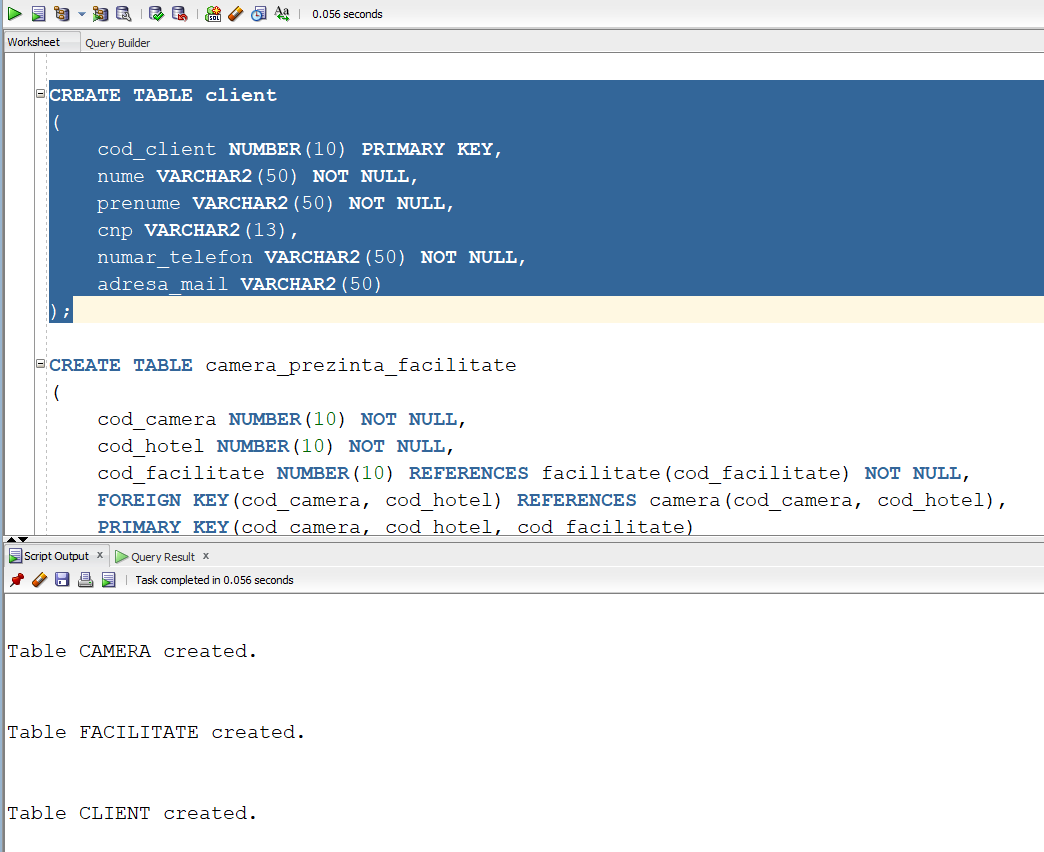


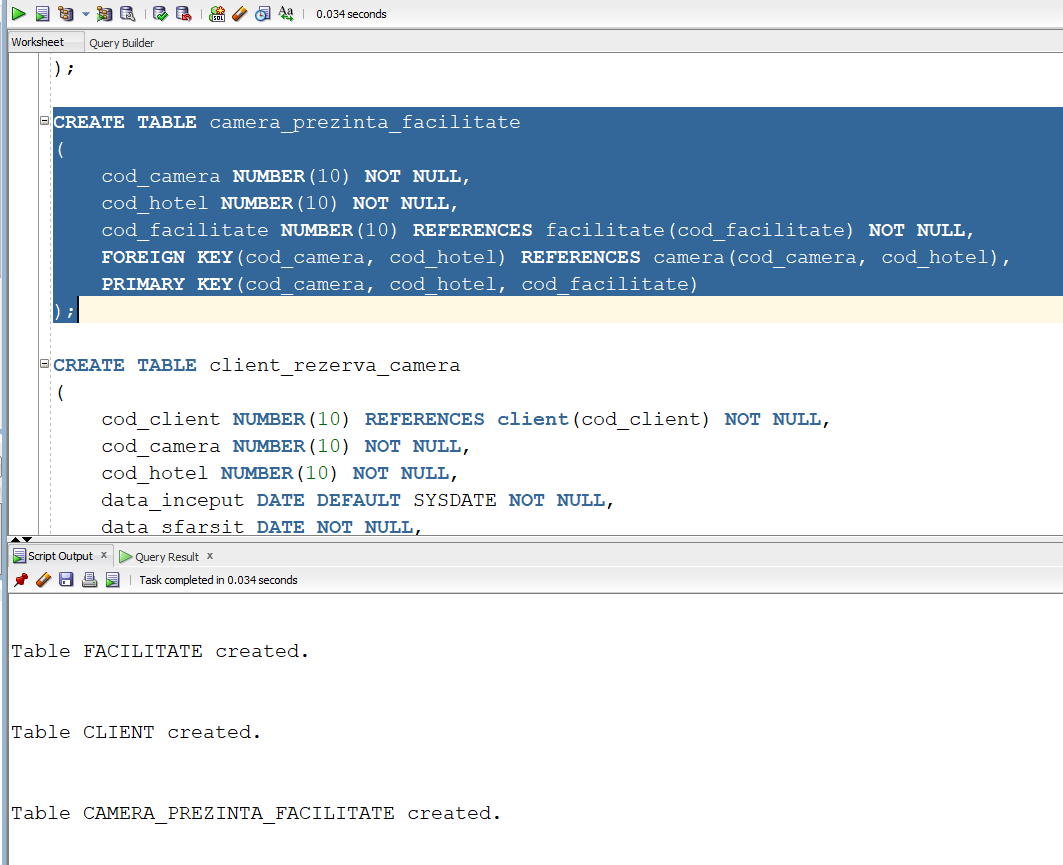


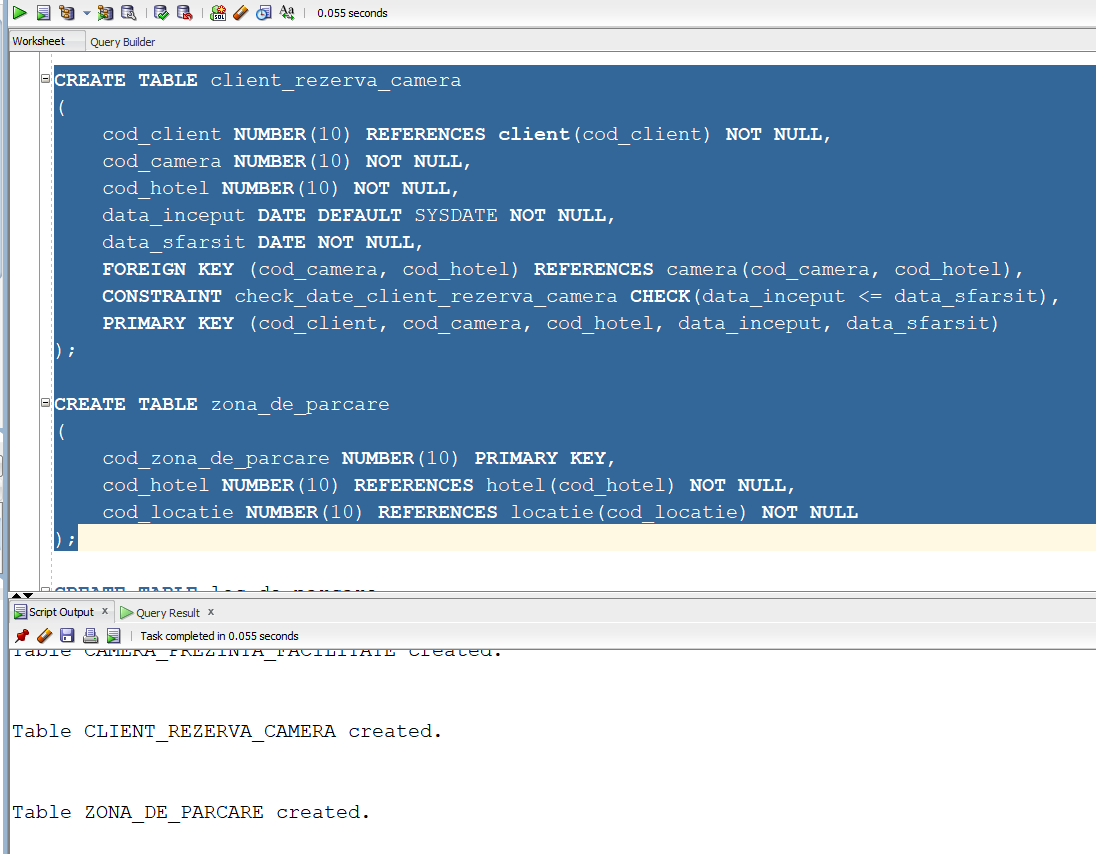


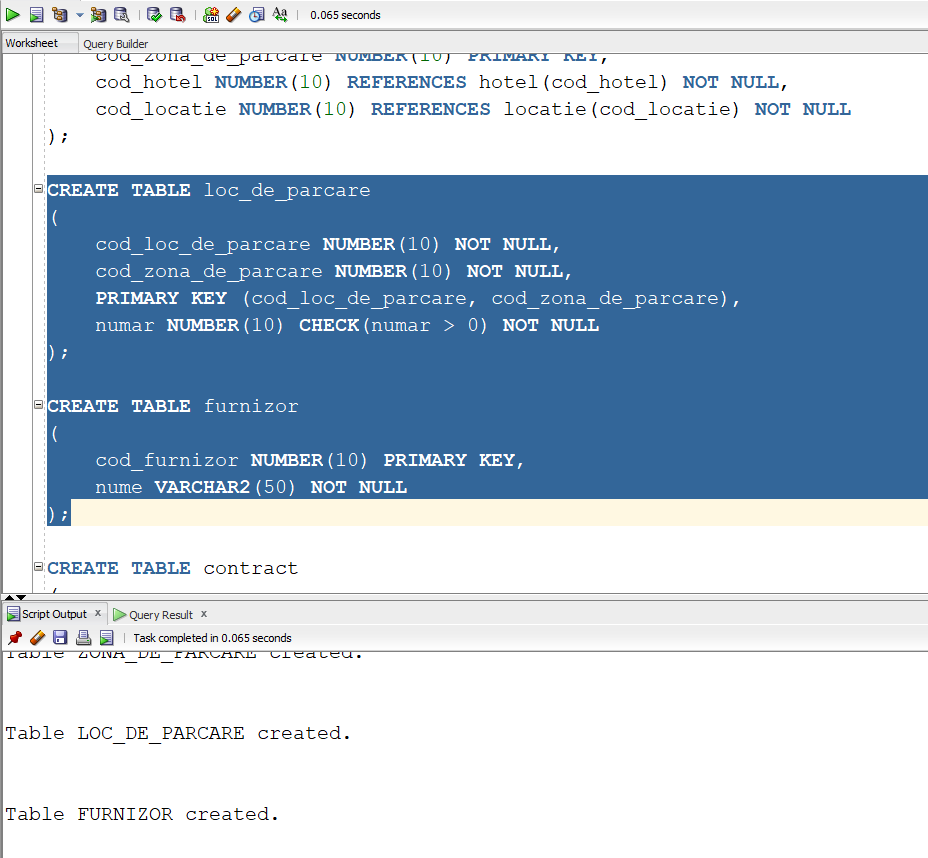


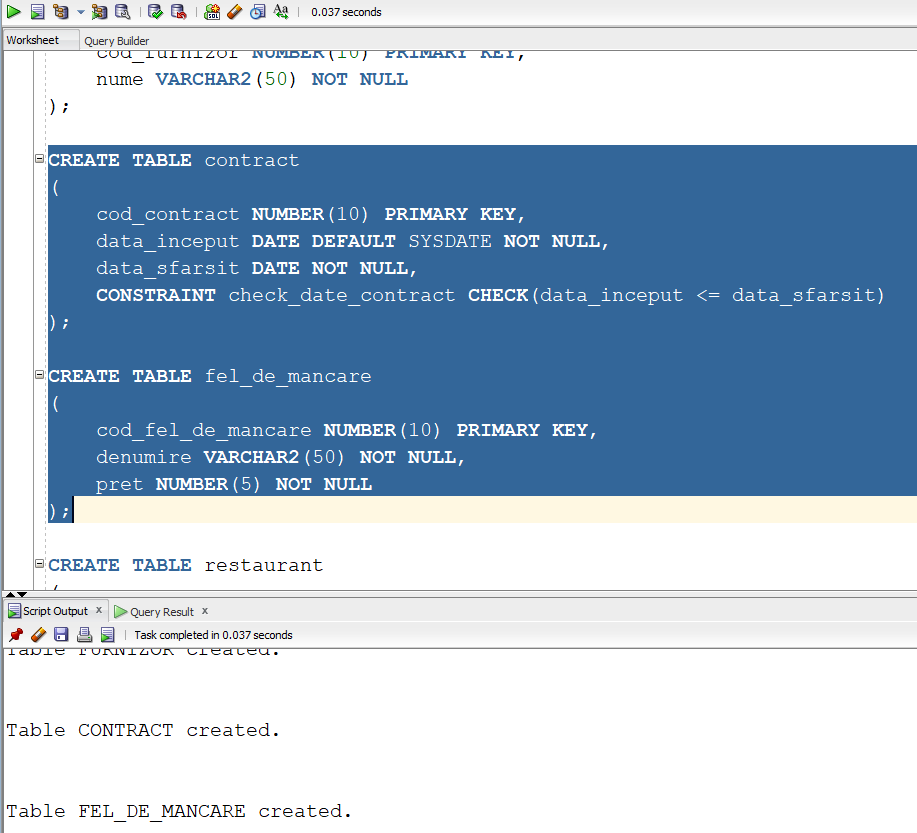


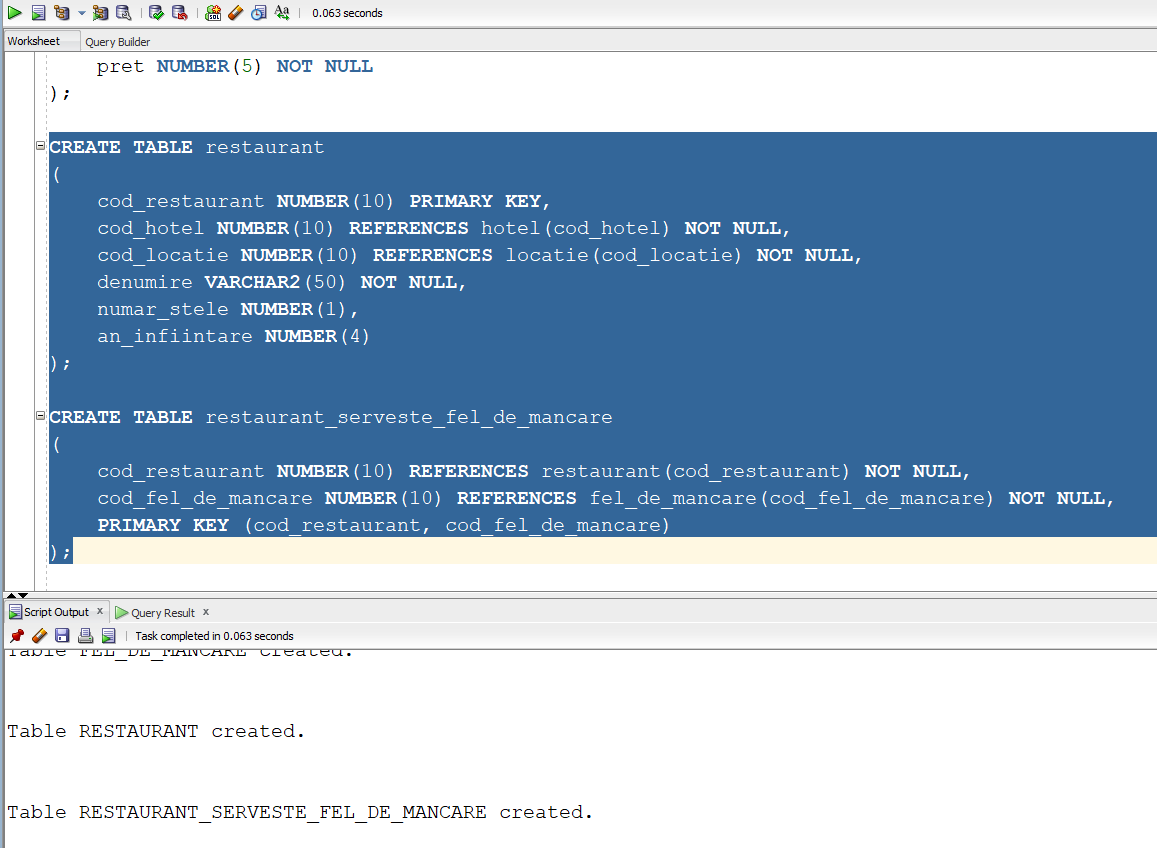


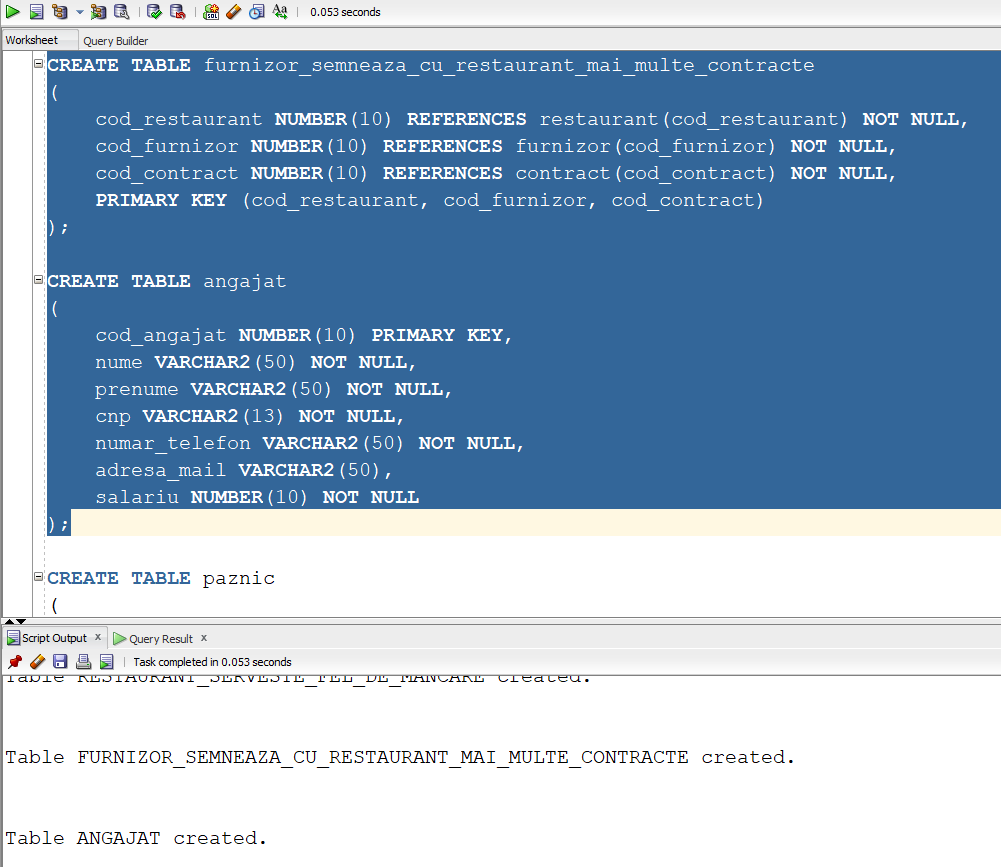


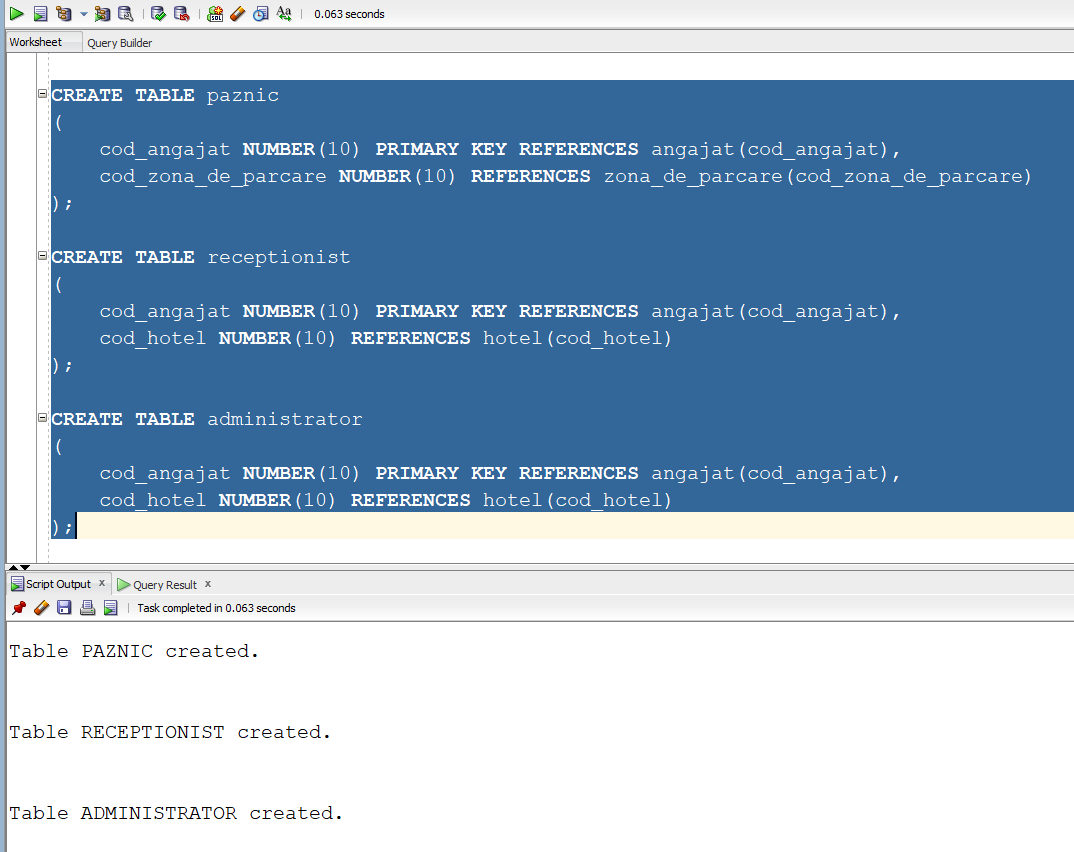


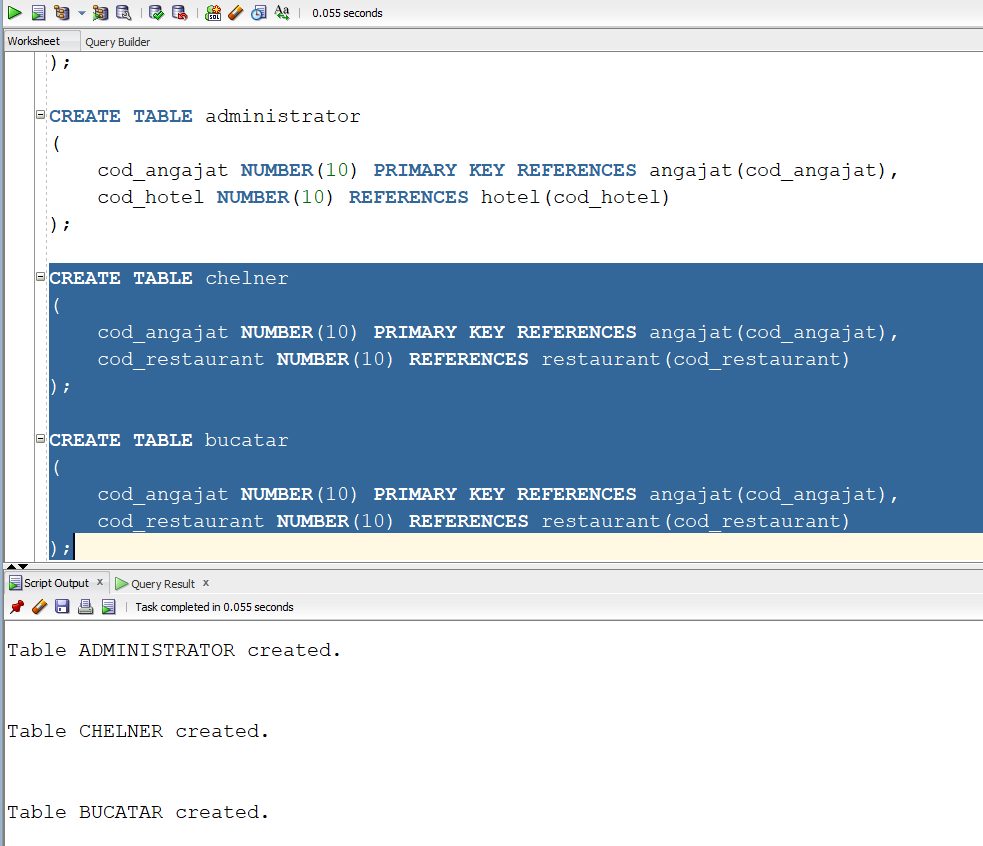






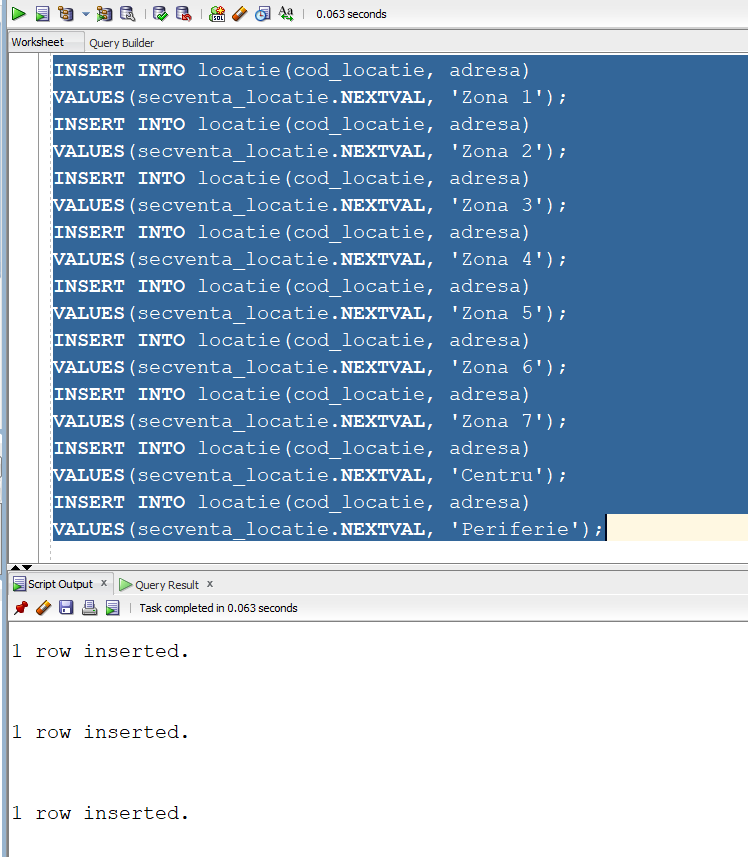




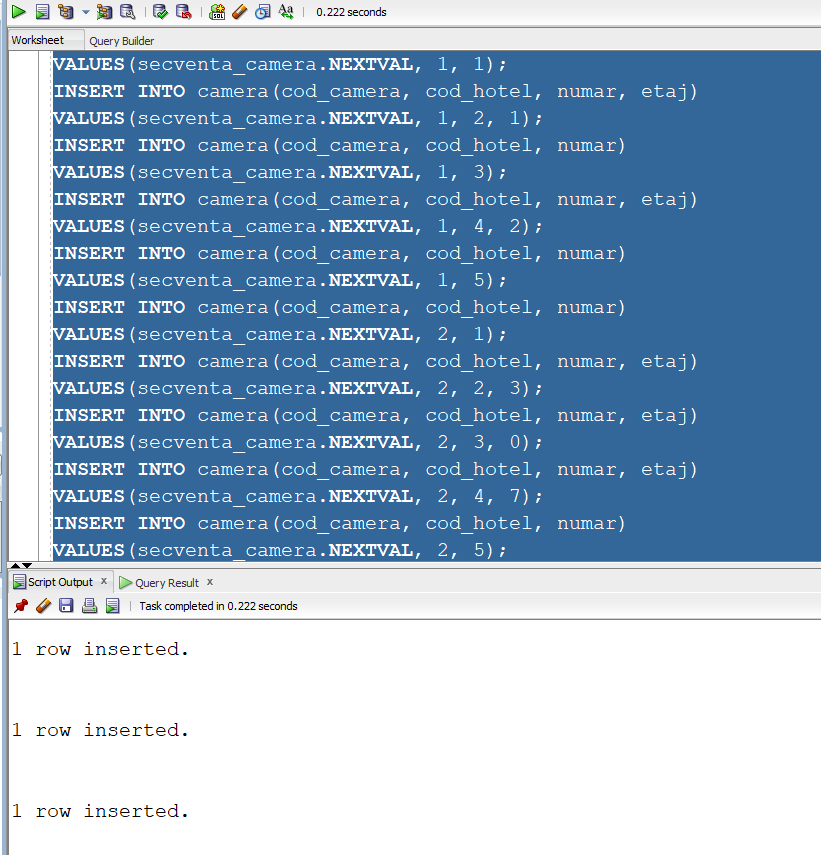


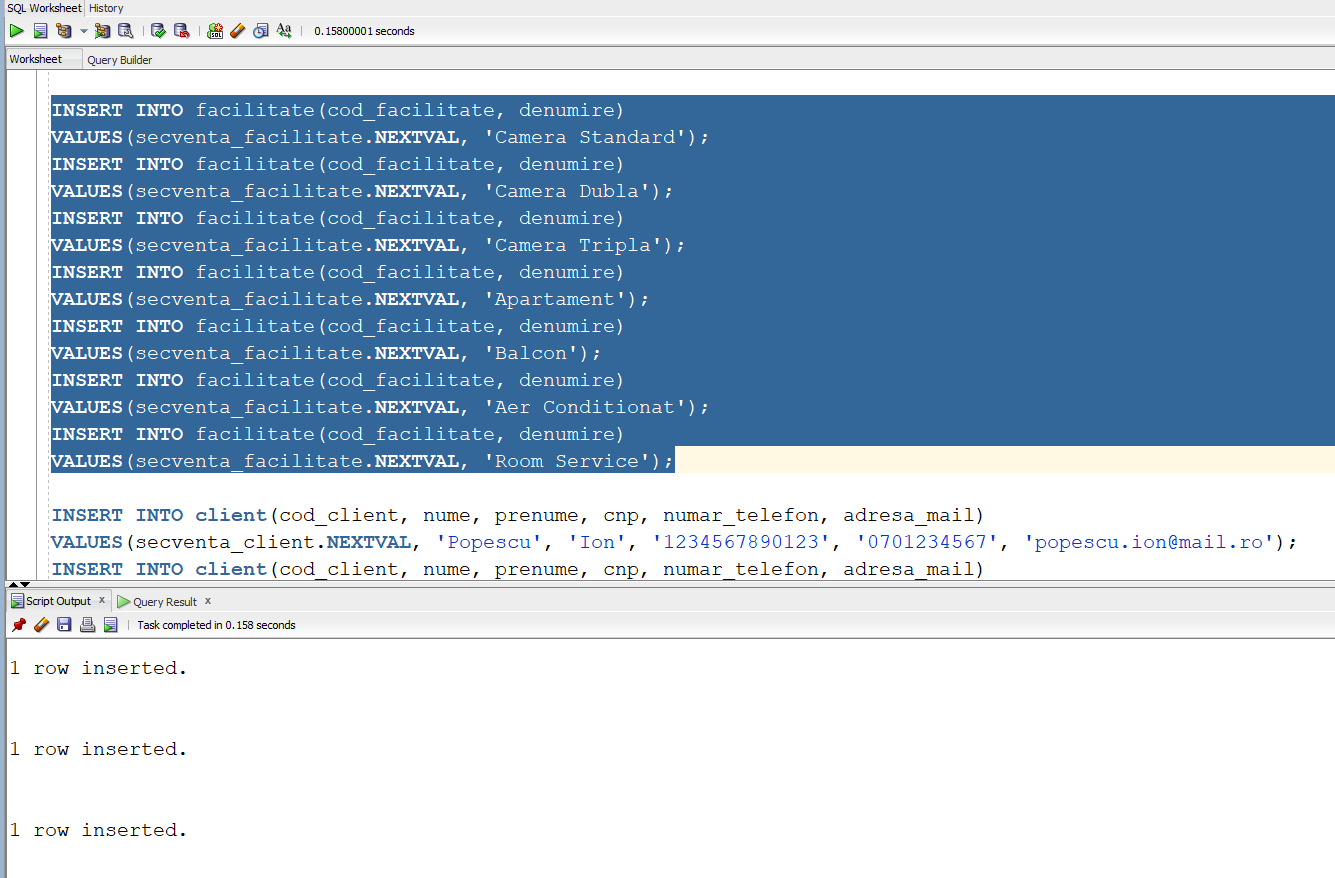
# Adaugarea informatiilor coerente in tabelele create

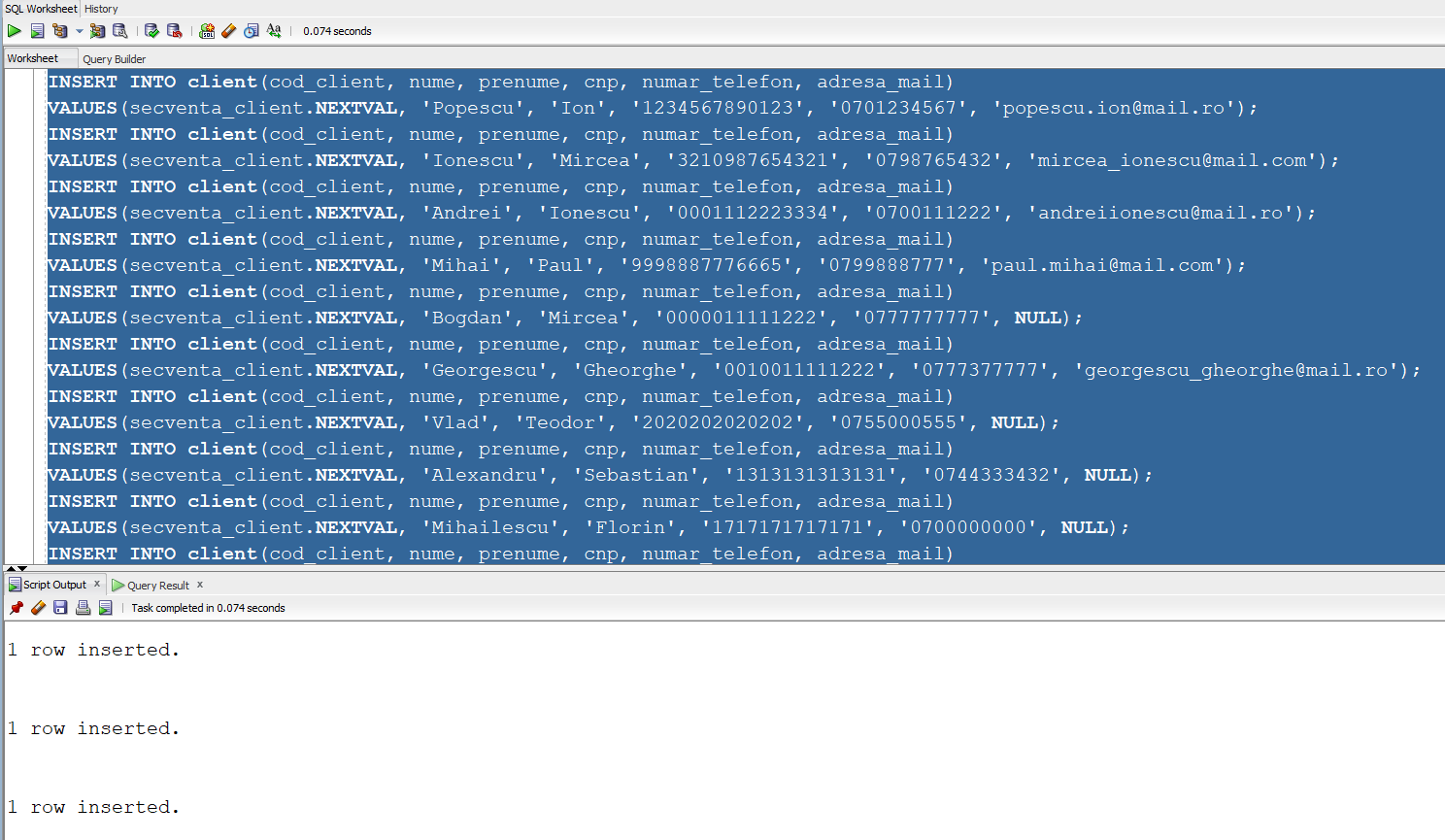
|  |
| --- |
| INSERT INTO locatie(cod\_locatie, adresa)  VALUES(secventa\_locatie.NEXTVAL, 'Zona 1');  INSERT INTO locatie(cod\_locatie, adresa)  VALUES(secventa\_locatie.NEXTVAL, 'Zona 2');  INSERT INTO locatie(cod\_locatie, adresa)  VALUES(secventa\_locatie.NEXTVAL, 'Zona 3');  INSERT INTO locatie(cod\_locatie, adresa)  VALUES(secventa\_locatie.NEXTVAL, 'Zona 4');  INSERT INTO locatie(cod\_locatie, adresa)  VALUES(secventa\_locatie.NEXTVAL, 'Zona 5');  INSERT INTO locatie(cod\_locatie, adresa)  VALUES(secventa\_locatie.NEXTVAL, 'Zona 6');  INSERT INTO locatie(cod\_locatie, adresa)  VALUES(secventa\_locatie.NEXTVAL, 'Zona 7');  INSERT INTO locatie(cod\_locatie, adresa)  VALUES(secventa\_locatie.NEXTVAL, 'Centru');  INSERT INTO locatie(cod\_locatie, adresa)  VALUES(secventa\_locatie.NEXTVAL, 'Periferie');    INSERT INTO hotel(cod\_hotel, denumire, numar\_stele, an\_infiintare, cod\_locatie)  VALUES(secventa\_hotel.NEXTVAL, 'Continental', 3, 1965, 1);  INSERT INTO hotel(cod\_hotel, denumire, numar\_stele, an\_infiintare, cod\_locatie)  VALUES(secventa\_hotel.NEXTVAL, 'Europa', 4, 1993, 1);  INSERT INTO hotel(cod\_hotel, denumire, numar\_stele, an\_infiintare, cod\_locatie)  VALUES(secventa\_hotel.NEXTVAL, 'Atena', 3, 1997, 1);  INSERT INTO hotel(cod\_hotel, denumire, numar\_stele, an\_infiintare, cod\_locatie)  VALUES(secventa\_hotel.NEXTVAL, 'Sofia', 2, 1983, 2);  INSERT INTO hotel(cod\_hotel, denumire, numar\_stele, an\_infiintare, cod\_locatie)  VALUES(secventa\_hotel.NEXTVAL, 'Mures', 4, 1975, 2);  INSERT INTO hotel(cod\_hotel, denumire, numar\_stele, an\_infiintare, cod\_locatie)  VALUES(secventa\_hotel.NEXTVAL, 'Banat', 2, 1968, 2);  INSERT INTO hotel(cod\_hotel, denumire, numar\_stele, an\_infiintare, cod\_locatie)  VALUES(secventa\_hotel.NEXTVAL, 'Transilvania', 3, 1970, 3);  INSERT INTO hotel(cod\_hotel, denumire, numar\_stele, an\_infiintare, cod\_locatie)  VALUES(secventa\_hotel.NEXTVAL, 'Moldova', NULL, NULL, 3);  INSERT INTO hotel(cod\_hotel, denumire, numar\_stele, an\_infiintare, cod\_locatie)  VALUES(secventa\_hotel.NEXTVAL, 'Dobrogea', 5, NULL, 9);  INSERT INTO hotel(cod\_hotel, denumire, numar\_stele, an\_infiintare, cod\_locatie)  VALUES(secventa\_hotel.NEXTVAL, 'Basarabia', 1, 1950, 9);  INSERT INTO hotel(cod\_hotel, denumire, numar\_stele, an\_infiintare, cod\_locatie)  VALUES(secventa\_hotel.NEXTVAL, 'Bucovina', NULL, NULL, 9);  INSERT INTO hotel(cod\_hotel, denumire, numar\_stele, an\_infiintare, cod\_locatie)  VALUES(secventa\_hotel.NEXTVAL, 'Transnistria', NULL, NULL, 8);  INSERT INTO hotel(cod\_hotel, denumire, numar\_stele, an\_infiintare, cod\_locatie)  VALUES(secventa\_hotel.NEXTVAL, 'Venetia', NULL, 1963, 8);  INSERT INTO hotel(cod\_hotel, denumire, numar\_stele, an\_infiintare, cod\_locatie)  VALUES(secventa\_hotel.NEXTVAL, 'Dunare', 3, NULL, 8);  INSERT INTO hotel(cod\_hotel, denumire, numar\_stele, an\_infiintare, cod\_locatie)  VALUES(secventa\_hotel.NEXTVAL, 'Carpati', NULL, 1968, 6);    INSERT INTO camera(cod\_camera, cod\_hotel, numar)  VALUES(secventa\_camera.NEXTVAL, 1, 1);  INSERT INTO camera(cod\_camera, cod\_hotel, numar, etaj)  VALUES(secventa\_camera.NEXTVAL, 1, 2, 1);  INSERT INTO camera(cod\_camera, cod\_hotel, numar)  VALUES(secventa\_camera.NEXTVAL, 1, 3);  INSERT INTO camera(cod\_camera, cod\_hotel, numar, etaj)  VALUES(secventa\_camera.NEXTVAL, 1, 4, 2);  INSERT INTO camera(cod\_camera, cod\_hotel, numar)  VALUES(secventa\_camera.NEXTVAL, 1, 5);  INSERT INTO camera(cod\_camera, cod\_hotel, numar)  VALUES(secventa\_camera.NEXTVAL, 2, 1);  INSERT INTO camera(cod\_camera, cod\_hotel, numar, etaj)  VALUES(secventa\_camera.NEXTVAL, 2, 2, 3);  INSERT INTO camera(cod\_camera, cod\_hotel, numar, etaj)  VALUES(secventa\_camera.NEXTVAL, 2, 3, 0);  INSERT INTO camera(cod\_camera, cod\_hotel, numar, etaj)  VALUES(secventa\_camera.NEXTVAL, 2, 4, 7);  INSERT INTO camera(cod\_camera, cod\_hotel, numar)  VALUES(secventa\_camera.NEXTVAL, 2, 5);  INSERT INTO camera(cod\_camera, cod\_hotel, numar, etaj)  VALUES(secventa\_camera.NEXTVAL, 3, 1, 1);    INSERT INTO facilitate(cod\_facilitate, denumire)  VALUES(secventa\_facilitate.NEXTVAL, 'Camera Standard');  INSERT INTO facilitate(cod\_facilitate, denumire)  VALUES(secventa\_facilitate.NEXTVAL, 'Camera Dubla');  INSERT INTO facilitate(cod\_facilitate, denumire)  VALUES(secventa\_facilitate.NEXTVAL, 'Camera Tripla');  INSERT INTO facilitate(cod\_facilitate, denumire)  VALUES(secventa\_facilitate.NEXTVAL, 'Apartament');  INSERT INTO facilitate(cod\_facilitate, denumire)  VALUES(secventa\_facilitate.NEXTVAL, 'Balcon');  INSERT INTO facilitate(cod\_facilitate, denumire)  VALUES(secventa\_facilitate.NEXTVAL, 'Aer Conditionat');  INSERT INTO facilitate(cod\_facilitate, denumire)  VALUES(secventa\_facilitate.NEXTVAL, 'Room Service');    INSERT INTO client(cod\_client, nume, prenume, cnp, numar\_telefon, adresa\_mail)  VALUES(secventa\_client.NEXTVAL, 'Popescu', 'Ion', '1234567890123', '0701234567', 'popescu.ion@mail.ro');  INSERT INTO client(cod\_client, nume, prenume, cnp, numar\_telefon, adresa\_mail)  VALUES(secventa\_client.NEXTVAL, 'Ionescu', 'Mircea', '3210987654321', '0798765432', 'mircea\_ionescu@mail.com');  INSERT INTO client(cod\_client, nume, prenume, cnp, numar\_telefon, adresa\_mail)  VALUES(secventa\_client.NEXTVAL, 'Andrei', 'Ionescu', '0001112223334', '0700111222', 'andreiionescu@mail.ro');  INSERT INTO client(cod\_client, nume, prenume, cnp, numar\_telefon, adresa\_mail)  VALUES(secventa\_client.NEXTVAL, 'Mihai', 'Paul', '9998887776665', '0799888777', 'paul.mihai@mail.com');  INSERT INTO client(cod\_client, nume, prenume, cnp, numar\_telefon, adresa\_mail)  VALUES(secventa\_client.NEXTVAL, 'Bogdan', 'Mircea', '0000011111222', '0777777777', NULL);  INSERT INTO client(cod\_client, nume, prenume, cnp, numar\_telefon, adresa\_mail)  VALUES(secventa\_client.NEXTVAL, 'Georgescu', 'Gheorghe', '0010011111222', '0777377777', 'georgescu\_gheorghe@mail.ro');  INSERT INTO client(cod\_client, nume, prenume, cnp, numar\_telefon, adresa\_mail)  VALUES(secventa\_client.NEXTVAL, 'Vlad', 'Teodor', '2020202020202', '0755000555', NULL);  INSERT INTO client(cod\_client, nume, prenume, cnp, numar\_telefon, adresa\_mail)  VALUES(secventa\_client.NEXTVAL, 'Alexandru', 'Sebastian', '1313131313131', '0744333432', NULL);  INSERT INTO client(cod\_client, nume, prenume, cnp, numar\_telefon, adresa\_mail)  VALUES(secventa\_client.NEXTVAL, 'Mihailescu', 'Florin', '1717171717171', '0700000000', NULL);  INSERT INTO client(cod\_client, nume, prenume, cnp, numar\_telefon, adresa\_mail)  VALUES(secventa\_client.NEXTVAL, 'Andreescu', 'Robert', '1919191919191', '0711222333', NULL);    INSERT INTO camera\_prezinta\_facilitate(cod\_camera, cod\_hotel, cod\_facilitate)  VALUES(1, 1, 1);  INSERT INTO camera\_prezinta\_facilitate(cod\_camera, cod\_hotel, cod\_facilitate)  VALUES(1, 1, 6);  INSERT INTO camera\_prezinta\_facilitate(cod\_camera, cod\_hotel, cod\_facilitate)  VALUES(1, 1, 7);  INSERT INTO camera\_prezinta\_facilitate(cod\_camera, cod\_hotel, cod\_facilitate)  VALUES(9, 2, 3);  INSERT INTO camera\_prezinta\_facilitate(cod\_camera, cod\_hotel, cod\_facilitate)  VALUES(9, 2, 5);  INSERT INTO camera\_prezinta\_facilitate(cod\_camera, cod\_hotel, cod\_facilitate)  VALUES(9, 2, 6);  INSERT INTO camera\_prezinta\_facilitate(cod\_camera, cod\_hotel, cod\_facilitate)  VALUES(9, 2, 7);  INSERT INTO camera\_prezinta\_facilitate(cod\_camera, cod\_hotel, cod\_facilitate)  VALUES(10, 2, 4);  INSERT INTO camera\_prezinta\_facilitate(cod\_camera, cod\_hotel, cod\_facilitate)  VALUES(10, 2, 5);  INSERT INTO camera\_prezinta\_facilitate(cod\_camera, cod\_hotel, cod\_facilitate)  VALUES(10, 2, 6);  INSERT INTO camera\_prezinta\_facilitate(cod\_camera, cod\_hotel, cod\_facilitate)  VALUES(8, 2, 2);  INSERT INTO camera\_prezinta\_facilitate(cod\_camera, cod\_hotel, cod\_facilitate)  VALUES(8, 2, 7);  INSERT INTO camera\_prezinta\_facilitate(cod\_camera, cod\_hotel, cod\_facilitate)  VALUES(7, 2, 7);  INSERT INTO camera\_prezinta\_facilitate(cod\_camera, cod\_hotel, cod\_facilitate)  VALUES(7, 2, 6);  INSERT INTO camera\_prezinta\_facilitate(cod\_camera, cod\_hotel, cod\_facilitate)  VALUES(7, 2, 1);    INSERT INTO client\_rezerva\_camera(cod\_client, cod\_camera, cod\_hotel, data\_inceput, data\_sfarsit)  VALUES(1, 1, 1, TO\_DATE('02-09-2023', 'dd-mm-yyyy'), TO\_DATE('08-09-2023', 'dd-mm-yyyy'));  INSERT INTO client\_rezerva\_camera(cod\_client, cod\_camera, cod\_hotel, data\_inceput, data\_sfarsit)  VALUES(2, 2, 1, TO\_DATE('07-05-2023', 'dd-mm-yyyy'), TO\_DATE('13-05-2023', 'dd-mm-yyyy'));  INSERT INTO client\_rezerva\_camera(cod\_client, cod\_camera, cod\_hotel, data\_inceput, data\_sfarsit)  VALUES(3, 3, 1, TO\_DATE('26-04-2023', 'dd-mm-yyyy'), TO\_DATE('02-05-2023', 'dd-mm-yyyy'));  INSERT INTO client\_rezerva\_camera(cod\_client, cod\_camera, cod\_hotel, data\_inceput, data\_sfarsit)  VALUES(4, 4, 1, TO\_DATE('23-03-2023', 'dd-mm-yyyy'), TO\_DATE('01-04-2023', 'dd-mm-yyyy'));  INSERT INTO client\_rezerva\_camera(cod\_client, cod\_camera, cod\_hotel, data\_inceput, data\_sfarsit)  VALUES(5, 5, 1, TO\_DATE('23-10-2023', 'dd-mm-yyyy'), TO\_DATE('01-11-2023', 'dd-mm-yyyy'));  INSERT INTO client\_rezerva\_camera(cod\_client, cod\_camera, cod\_hotel, data\_inceput, data\_sfarsit)  VALUES(5, 7, 2, TO\_DATE('07-09-2023', 'dd-mm-yyyy'), TO\_DATE('08-09-2023', 'dd-mm-yyyy'));  INSERT INTO client\_rezerva\_camera(cod\_client, cod\_camera, cod\_hotel, data\_inceput, data\_sfarsit)  VALUES(6, 7, 2, TO\_DATE('10-03-2023', 'dd-mm-yyyy'), TO\_DATE('14-03-2023', 'dd-mm-yyyy'));  INSERT INTO client\_rezerva\_camera(cod\_client, cod\_camera, cod\_hotel, data\_inceput, data\_sfarsit)  VALUES(6, 8, 2, TO\_DATE('15-04-2023', 'dd-mm-yyyy'), TO\_DATE('15-05-2023', 'dd-mm-yyyy'));  INSERT INTO client\_rezerva\_camera(cod\_client, cod\_camera, cod\_hotel, data\_inceput, data\_sfarsit)  VALUES(9, 9, 2, TO\_DATE('17-03-2023', 'dd-mm-yyyy'), TO\_DATE('19-05-2023', 'dd-mm-yyyy'));  INSERT INTO client\_rezerva\_camera(cod\_client, cod\_camera, cod\_hotel, data\_inceput, data\_sfarsit)  VALUES(9, 9, 2, TO\_DATE('01-01-2023', 'dd-mm-yyyy'), TO\_DATE('01-02-2023', 'dd-mm-yyyy'));    INSERT INTO zona\_de\_parcare(cod\_zona\_de\_parcare, cod\_hotel, cod\_locatie)  VALUES(secventa\_zona\_de\_parcare.NEXTVAL, 1, 1);  INSERT INTO zona\_de\_parcare(cod\_zona\_de\_parcare, cod\_hotel, cod\_locatie)  VALUES(secventa\_zona\_de\_parcare.NEXTVAL, 2, 2);  INSERT INTO zona\_de\_parcare(cod\_zona\_de\_parcare, cod\_hotel, cod\_locatie)  VALUES(secventa\_zona\_de\_parcare.NEXTVAL, 3, 4);  INSERT INTO zona\_de\_parcare(cod\_zona\_de\_parcare, cod\_hotel, cod\_locatie)  VALUES(secventa\_zona\_de\_parcare.NEXTVAL, 1, 1);  INSERT INTO zona\_de\_parcare(cod\_zona\_de\_parcare, cod\_hotel, cod\_locatie)  VALUES(secventa\_zona\_de\_parcare.NEXTVAL, 4, 3);  INSERT INTO zona\_de\_parcare(cod\_zona\_de\_parcare, cod\_hotel, cod\_locatie)  VALUES(secventa\_zona\_de\_parcare.NEXTVAL, 5, 5);  INSERT INTO zona\_de\_parcare(cod\_zona\_de\_parcare, cod\_hotel, cod\_locatie)  VALUES(secventa\_zona\_de\_parcare.NEXTVAL, 6, 5);  INSERT INTO zona\_de\_parcare(cod\_zona\_de\_parcare, cod\_hotel, cod\_locatie)  VALUES(secventa\_zona\_de\_parcare.NEXTVAL, 1, 5);  INSERT INTO zona\_de\_parcare(cod\_zona\_de\_parcare, cod\_hotel, cod\_locatie)  VALUES(secventa\_zona\_de\_parcare.NEXTVAL, 7, 6);  INSERT INTO zona\_de\_parcare(cod\_zona\_de\_parcare, cod\_hotel, cod\_locatie)  VALUES(secventa\_zona\_de\_parcare.NEXTVAL, 8, 6);    INSERT INTO loc\_de\_parcare(cod\_loc\_de\_parcare, cod\_zona\_de\_parcare, numar)  VALUES(secventa\_loc\_de\_parcare.NEXTVAL, 1, 1);  INSERT INTO loc\_de\_parcare(cod\_loc\_de\_parcare, cod\_zona\_de\_parcare, numar)  VALUES(secventa\_loc\_de\_parcare.NEXTVAL, 1, 2);  INSERT INTO loc\_de\_parcare(cod\_loc\_de\_parcare, cod\_zona\_de\_parcare, numar)  VALUES(secventa\_loc\_de\_parcare.NEXTVAL, 1, 3);  INSERT INTO loc\_de\_parcare(cod\_loc\_de\_parcare, cod\_zona\_de\_parcare, numar)  VALUES(secventa\_loc\_de\_parcare.NEXTVAL, 1, 4);  INSERT INTO loc\_de\_parcare(cod\_loc\_de\_parcare, cod\_zona\_de\_parcare, numar)  VALUES(secventa\_loc\_de\_parcare.NEXTVAL, 1, 5);  INSERT INTO loc\_de\_parcare(cod\_loc\_de\_parcare, cod\_zona\_de\_parcare, numar)  VALUES(secventa\_loc\_de\_parcare.NEXTVAL, 1, 6);  INSERT INTO loc\_de\_parcare(cod\_loc\_de\_parcare, cod\_zona\_de\_parcare, numar)  VALUES(secventa\_loc\_de\_parcare.NEXTVAL, 2, 1);  INSERT INTO loc\_de\_parcare(cod\_loc\_de\_parcare, cod\_zona\_de\_parcare, numar)  VALUES(secventa\_loc\_de\_parcare.NEXTVAL, 2, 2);  INSERT INTO loc\_de\_parcare(cod\_loc\_de\_parcare, cod\_zona\_de\_parcare, numar)  VALUES(secventa\_loc\_de\_parcare.NEXTVAL, 2, 3);  INSERT INTO loc\_de\_parcare(cod\_loc\_de\_parcare, cod\_zona\_de\_parcare, numar)  VALUES(secventa\_loc\_de\_parcare.NEXTVAL, 2, 4);  INSERT INTO loc\_de\_parcare(cod\_loc\_de\_parcare, cod\_zona\_de\_parcare, numar)  VALUES(secventa\_loc\_de\_parcare.NEXTVAL, 3, 1);  INSERT INTO loc\_de\_parcare(cod\_loc\_de\_parcare, cod\_zona\_de\_parcare, numar)  VALUES(secventa\_loc\_de\_parcare.NEXTVAL, 3, 2);  INSERT INTO loc\_de\_parcare(cod\_loc\_de\_parcare, cod\_zona\_de\_parcare, numar)  VALUES(secventa\_loc\_de\_parcare.NEXTVAL, 3, 3);  INSERT INTO loc\_de\_parcare(cod\_loc\_de\_parcare, cod\_zona\_de\_parcare, numar)  VALUES(secventa\_loc\_de\_parcare.NEXTVAL, 3, 4);  INSERT INTO loc\_de\_parcare(cod\_loc\_de\_parcare, cod\_zona\_de\_parcare, numar)  VALUES(secventa\_loc\_de\_parcare.NEXTVAL, 3, 5);    INSERT INTO furnizor(cod\_furnizor, nume)  VALUES(secventa\_furnizor.NEXTVAL, 'Brutaria Iberia S.R.L.');  INSERT INTO furnizor(cod\_furnizor, nume)  VALUES(secventa\_furnizor.NEXTVAL, 'Macelaria Creta S.R.L.');  INSERT INTO furnizor(cod\_furnizor, nume)  VALUES(secventa\_furnizor.NEXTVAL, 'Apa Dorna S.R.L.');  INSERT INTO furnizor(cod\_furnizor, nume)  VALUES(secventa\_furnizor.NEXTVAL, 'Brutaria Pavel S.R.L.');  INSERT INTO furnizor(cod\_furnizor, nume)  VALUES(secventa\_furnizor.NEXTVAL, 'Macelarie Centru S.R.L.');  INSERT INTO furnizor(cod\_furnizor, nume)  VALUES(secventa\_furnizor.NEXTVAL, 'Apa Carpatica S.R.L.');    INSERT INTO contract(cod\_contract, data\_inceput, data\_sfarsit)  VALUES(secventa\_contract.NEXTVAL, TO\_DATE('02-09-2023', 'dd-mm-yyyy'), TO\_DATE('02-12-2023', 'dd-mm-yyyy'));  INSERT INTO contract(cod\_contract, data\_inceput, data\_sfarsit)  VALUES(secventa\_contract.NEXTVAL, TO\_DATE('01-01-2024', 'dd-mm-yyyy'), TO\_DATE('01-01-2025', 'dd-mm-yyyy'));  INSERT INTO contract(cod\_contract, data\_inceput, data\_sfarsit)  VALUES(secventa\_contract.NEXTVAL, TO\_DATE('01-01-2023', 'dd-mm-yyyy'), TO\_DATE('01-07-2023', 'dd-mm-yyyy'));  INSERT INTO contract(cod\_contract, data\_inceput, data\_sfarsit)  VALUES(secventa\_contract.NEXTVAL, TO\_DATE('01-01-2023', 'dd-mm-yyyy'), TO\_DATE('01-08-2023', 'dd-mm-yyyy'));  INSERT INTO contract(cod\_contract, data\_inceput, data\_sfarsit)  VALUES(secventa\_contract.NEXTVAL, TO\_DATE('01-09-2023', 'dd-mm-yyyy'), TO\_DATE('01-01-2024', 'dd-mm-yyyy'));  INSERT INTO contract(cod\_contract, data\_inceput, data\_sfarsit)  VALUES(secventa\_contract.NEXTVAL, TO\_DATE('01-03-2023', 'dd-mm-yyyy'), TO\_DATE('01-04-2023', 'dd-mm-yyyy'));  INSERT INTO contract(cod\_contract, data\_inceput, data\_sfarsit)  VALUES(secventa\_contract.NEXTVAL, TO\_DATE('01-04-2023', 'dd-mm-yyyy'), TO\_DATE('12-05-2023', 'dd-mm-yyyy'));  INSERT INTO contract(cod\_contract, data\_inceput, data\_sfarsit)  VALUES(secventa\_contract.NEXTVAL, TO\_DATE('26-04-2023', 'dd-mm-yyyy'), TO\_DATE('02-06-2023', 'dd-mm-yyyy'));  INSERT INTO contract(cod\_contract, data\_inceput, data\_sfarsit)  VALUES(secventa\_contract.NEXTVAL, TO\_DATE('01-09-2023', 'dd-mm-yyyy'), TO\_DATE('01-01-2025', 'dd-mm-yyyy'));    INSERT INTO fel\_de\_mancare(cod\_fel\_de\_mancare, denumire, pret)  VALUES(secventa\_fel\_de\_mancare.NEXTVAL, 'Ciorba De Legume', 11);  INSERT INTO fel\_de\_mancare(cod\_fel\_de\_mancare, denumire, pret)  VALUES(secventa\_fel\_de\_mancare.NEXTVAL, 'Ciorba De Perisoare', 12);  INSERT INTO fel\_de\_mancare(cod\_fel\_de\_mancare, denumire, pret)  VALUES(secventa\_fel\_de\_mancare.NEXTVAL, 'Ciorba De Pui', 13);  INSERT INTO fel\_de\_mancare(cod\_fel\_de\_mancare, denumire, pret)  VALUES(secventa\_fel\_de\_mancare.NEXTVAL, 'Supa De Legume', 7);  INSERT INTO fel\_de\_mancare(cod\_fel\_de\_mancare, denumire, pret)  VALUES(secventa\_fel\_de\_mancare.NEXTVAL, 'Cartofi Prajiti', 15);  INSERT INTO fel\_de\_mancare(cod\_fel\_de\_mancare, denumire, pret)  VALUES(secventa\_fel\_de\_mancare.NEXTVAL, 'Piure', 12);  INSERT INTO fel\_de\_mancare(cod\_fel\_de\_mancare, denumire, pret)  VALUES(secventa\_fel\_de\_mancare.NEXTVAL, 'Spaghete Carbonara', 22);  INSERT INTO fel\_de\_mancare(cod\_fel\_de\_mancare, denumire, pret)  VALUES(secventa\_fel\_de\_mancare.NEXTVAL, 'Pizza', 27);  INSERT INTO fel\_de\_mancare(cod\_fel\_de\_mancare, denumire, pret)  VALUES(secventa\_fel\_de\_mancare.NEXTVAL, 'Limonada', 7);  INSERT INTO fel\_de\_mancare(cod\_fel\_de\_mancare, denumire, pret)  VALUES(secventa\_fel\_de\_mancare.NEXTVAL, 'Somon', 24);  INSERT INTO fel\_de\_mancare(cod\_fel\_de\_mancare, denumire, pret)  VALUES(secventa\_fel\_de\_mancare.NEXTVAL, 'Snitel De Pui', 14);  INSERT INTO fel\_de\_mancare(cod\_fel\_de\_mancare, denumire, pret)  VALUES(secventa\_fel\_de\_mancare.NEXTVAL, 'Coca Cola', 7);  INSERT INTO fel\_de\_mancare(cod\_fel\_de\_mancare, denumire, pret)  VALUES(secventa\_fel\_de\_mancare.NEXTVAL, 'Pepsi', 7);  INSERT INTO fel\_de\_mancare(cod\_fel\_de\_mancare, denumire, pret)  VALUES(secventa\_fel\_de\_mancare.NEXTVAL, 'Sprite', 7);  INSERT INTO fel\_de\_mancare(cod\_fel\_de\_mancare, denumire, pret)  VALUES(secventa\_fel\_de\_mancare.NEXTVAL, 'Fanta', 7);    INSERT INTO restaurant(cod\_restaurant, cod\_hotel, cod\_locatie, denumire, numar\_stele, an\_infiintare)  VALUES(secventa\_restaurant.NEXTVAL, 1, 5, 'Roma', 3, 1995);  INSERT INTO restaurant(cod\_restaurant, cod\_hotel, cod\_locatie, denumire, numar\_stele, an\_infiintare)  VALUES(secventa\_restaurant.NEXTVAL, 1, 2, 'Milano', 2, NULL);  INSERT INTO restaurant(cod\_restaurant, cod\_hotel, cod\_locatie, denumire, numar\_stele, an\_infiintare)  VALUES(secventa\_restaurant.NEXTVAL, 1, 5, 'Savoia', NULL, 2000);  INSERT INTO restaurant(cod\_restaurant, cod\_hotel, cod\_locatie, denumire, numar\_stele, an\_infiintare)  VALUES(secventa\_restaurant.NEXTVAL, 7, 3, 'Varsovia', 4, 2004);  INSERT INTO restaurant(cod\_restaurant, cod\_hotel, cod\_locatie, denumire, numar\_stele, an\_infiintare)  VALUES(secventa\_restaurant.NEXTVAL, 1, 5, 'Napoli', NULL, 2008);  INSERT INTO restaurant(cod\_restaurant, cod\_hotel, cod\_locatie, denumire, numar\_stele, an\_infiintare)  VALUES(secventa\_restaurant.NEXTVAL, 4, 7, 'Madrid', NULL, 1993);  INSERT INTO restaurant(cod\_restaurant, cod\_hotel, cod\_locatie, denumire, numar\_stele, an\_infiintare)  VALUES(secventa\_restaurant.NEXTVAL, 1, 5, 'Paris', NULL, NULL);  INSERT INTO restaurant(cod\_restaurant, cod\_hotel, cod\_locatie, denumire, numar\_stele, an\_infiintare)  VALUES(secventa\_restaurant.NEXTVAL, 2, 1, 'Lisabona', 3, NULL);    INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(1, 1);  INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(1, 2);  INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(2, 3);  INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(2, 4);  INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(2, 5);  INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(1, 5);  INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(1, 7);  INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(3, 2);  INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(3, 5);  INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(4, 2);  INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(5, 2);  INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(6, 3);  INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(8, 4);  INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(1, 6);  INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(8, 6);  INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(1, 10);  INSERT INTO restaurant\_serveste\_fel\_de\_mancare(cod\_restaurant, cod\_fel\_de\_mancare)  VALUES(2, 8);    INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(1, 1, 1);  INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(1, 1, 2);  INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(1, 1, 3);  INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(1, 1, 4);  INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(2, 1, 5);  INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(2, 1, 6);  INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(2, 2, 1);  INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(2, 2, 6);  INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(3, 4, 1);  INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(3, 4, 2);  INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(3, 5, 3);  INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(3, 6, 4);  INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(3, 6, 5);  INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(3, 6, 6);  INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(4, 5, 5);  INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(4, 5, 6);    INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Popescu', 'Florin', '0000000000001', '0700000001', NULL, 4000);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Stoica', 'Marius', '0000000000002', '0700000002', 'stoicamarius@mail.com', 4100);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Stan', 'Ioan', '0000000000003', '0700000003', NULL, 4200);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Vasilescu', 'George', '0000000000004', '0700000004', 'vasilescugeorge@mail.ro', 5000);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Vasilescu', 'Adrian', '0000000000005', '0700000005', NULL, 4700);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Radu', 'Andrei', '0000000000006', '0700000006', NULL, 4500);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Marinescu', 'Cristian', '0000000000007', '0700000007', NULL, 4300);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Marinescu', 'Dumitru', '0000000000008', '0700000008', 'dumitru.marinescu@mail.com', 5300);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Marin', 'Iacob', '0000000000009', '0700000009', NULL, 5200);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Florinescu', 'Cristian', '0000000000010', '0700000010', NULL, 4900);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Popa', 'Ion', '0000000000011', '0700000011', NULL, 4950);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Lupu', 'Daria', '0000000000012', '0700000012', NULL, 5500);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Adamescu', 'Luca', '0000000000013', '0700000013', NULL, 5500);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Ionescu', 'Mirela', '0000000000014', '0700000014', NULL, 5250);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Paraschiv', 'Darius', '0000000000015', '0700000015', 'paraschivdarius@mail.com', 5150);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Serban', 'Stefan', '0000000000016', '0700000016', NULL, 5050);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Victor', 'Camelia', '0000000000017', '0700000017', 'camelia.victor@mail.ro', 4750);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Dan', 'Oana', '0000000000018', '0700000018', 'dan\_oana@mail.com', 4850);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Cosmin', 'Ciprian', '0000000000019', '0700000019', 'ciprian.cosmin@mail.ro', 5000);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Popescu', 'Iulian', '0000000000020', '0700000020', NULL, 5000);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Popescu', 'Lucian', '0000000000021', '0700000021', NULL, 5750);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Popescu', 'Raluca', '0000000000022', '0700000022', NULL, 6050);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Tudor', 'Serban', '0000000000023', '0700000023', 'serban.tudor@mail.com', 6350);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Octavian', 'Silviu', '0000000000024', '0700000024', NULL, 7050);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Valentin', 'Catalin', '0000000000025', '0700000025', NULL, 6750);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Emanuel', 'Anca', '0000000000026', '0700000026', 'emanuel.anca@mail.com', 6550);  INSERT INTO angajat(cod\_angajat, nume, prenume, cnp, numar\_telefon, adresa\_mail, salariu)  VALUES(secventa\_angajat.NEXTVAL, 'Sorin', 'Darius', '0000000000027', '0700000027', 'sorin.darius@mail.ro', 6350);      INSERT INTO paznic(cod\_angajat, cod\_zona\_de\_parcare)  VALUES(1, 1);  INSERT INTO paznic(cod\_angajat, cod\_zona\_de\_parcare)  VALUES(2, NULL);  INSERT INTO paznic(cod\_angajat, cod\_zona\_de\_parcare)  VALUES(3, 2);  INSERT INTO paznic(cod\_angajat, cod\_zona\_de\_parcare)  VALUES(4, 2);  INSERT INTO paznic(cod\_angajat, cod\_zona\_de\_parcare)  VALUES(5, 3);    INSERT INTO receptionist(cod\_angajat, cod\_hotel)  VALUES(6, NULL);  INSERT INTO receptionist(cod\_angajat, cod\_hotel)  VALUES(7, 2);  INSERT INTO receptionist(cod\_angajat, cod\_hotel)  VALUES(8, 1);  INSERT INTO receptionist(cod\_angajat, cod\_hotel)  VALUES(9, NULL);  INSERT INTO receptionist(cod\_angajat, cod\_hotel)  VALUES(10, 1);    INSERT INTO chelner(cod\_angajat, cod\_restaurant)  VALUES(11, 1);  INSERT INTO chelner(cod\_angajat, cod\_restaurant)  VALUES(12, 1);  INSERT INTO chelner(cod\_angajat, cod\_restaurant)  VALUES(13, 2);  INSERT INTO chelner(cod\_angajat, cod\_restaurant)  VALUES(14, 2);  INSERT INTO chelner(cod\_angajat, cod\_restaurant)  VALUES(15, 4);    INSERT INTO bucatar(cod\_angajat, cod\_restaurant)  VALUES(16, 1);  INSERT INTO bucatar(cod\_angajat, cod\_restaurant)  VALUES(17, NULL);  INSERT INTO bucatar(cod\_angajat, cod\_restaurant)  VALUES(18, 1);  INSERT INTO bucatar(cod\_angajat, cod\_restaurant)  VALUES(19, NULL);  INSERT INTO bucatar(cod\_angajat, cod\_restaurant)  VALUES(20, 1);    INSERT INTO administrator(cod\_angajat, cod\_hotel)  VALUES(21, 1);  INSERT INTO administrator(cod\_angajat, cod\_hotel)  VALUES(22, 2);  INSERT INTO administrator(cod\_angajat, cod\_hotel)  VALUES(23, 3);  INSERT INTO administrator(cod\_angajat, cod\_hotel)  VALUES(24, 4);  INSERT INTO administrator(cod\_angajat, cod\_hotel)  VALUES(25, NULL); |

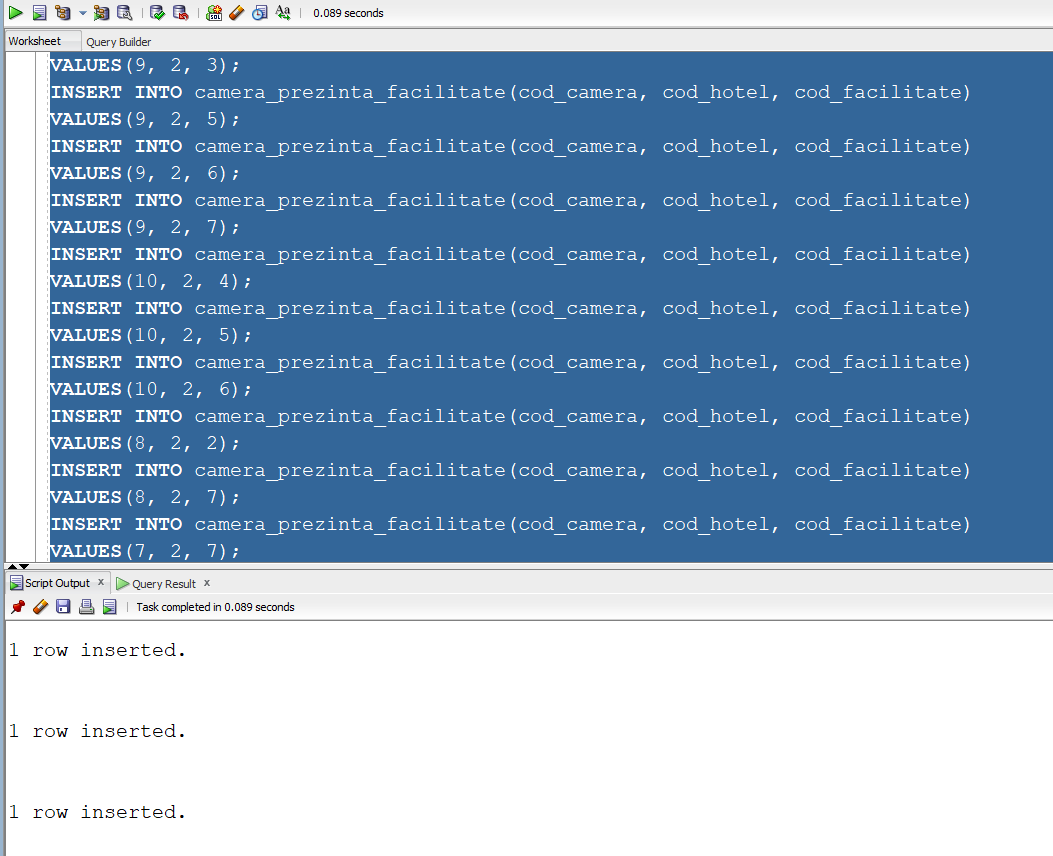


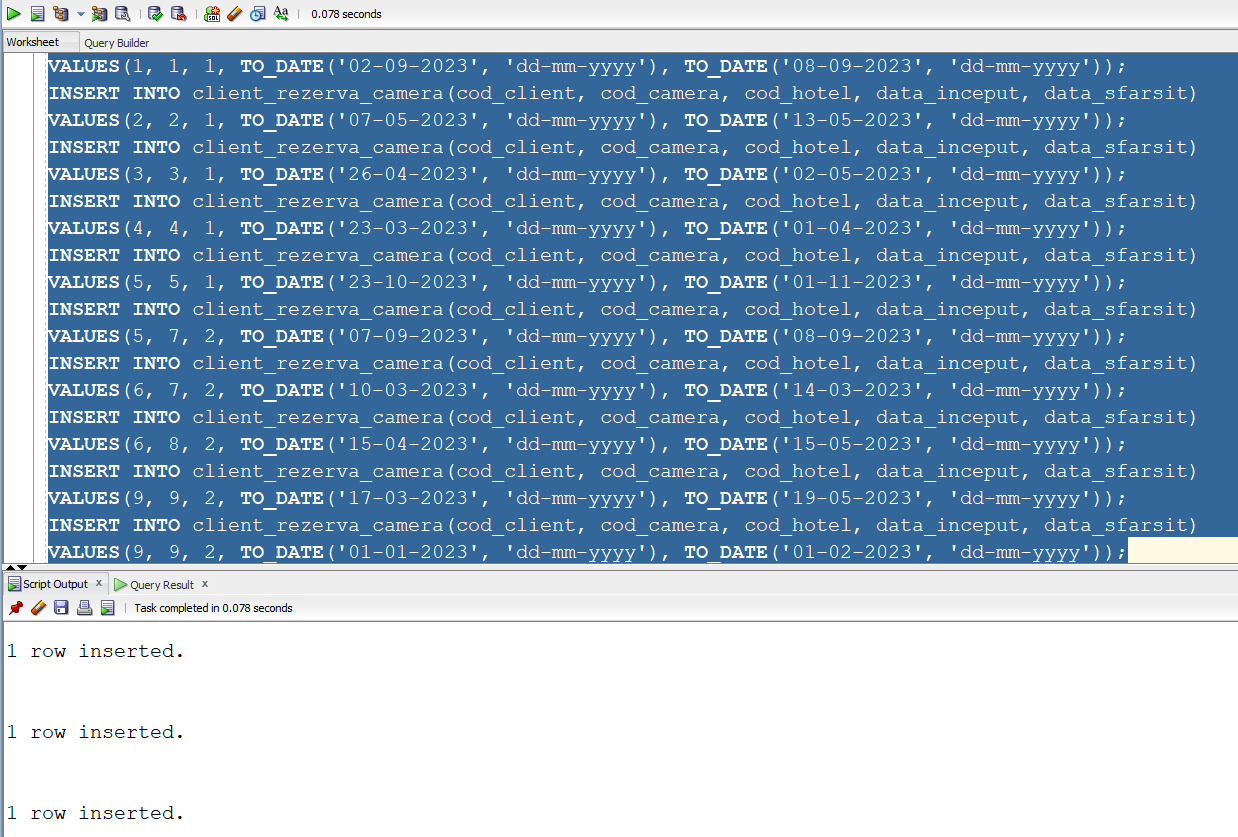


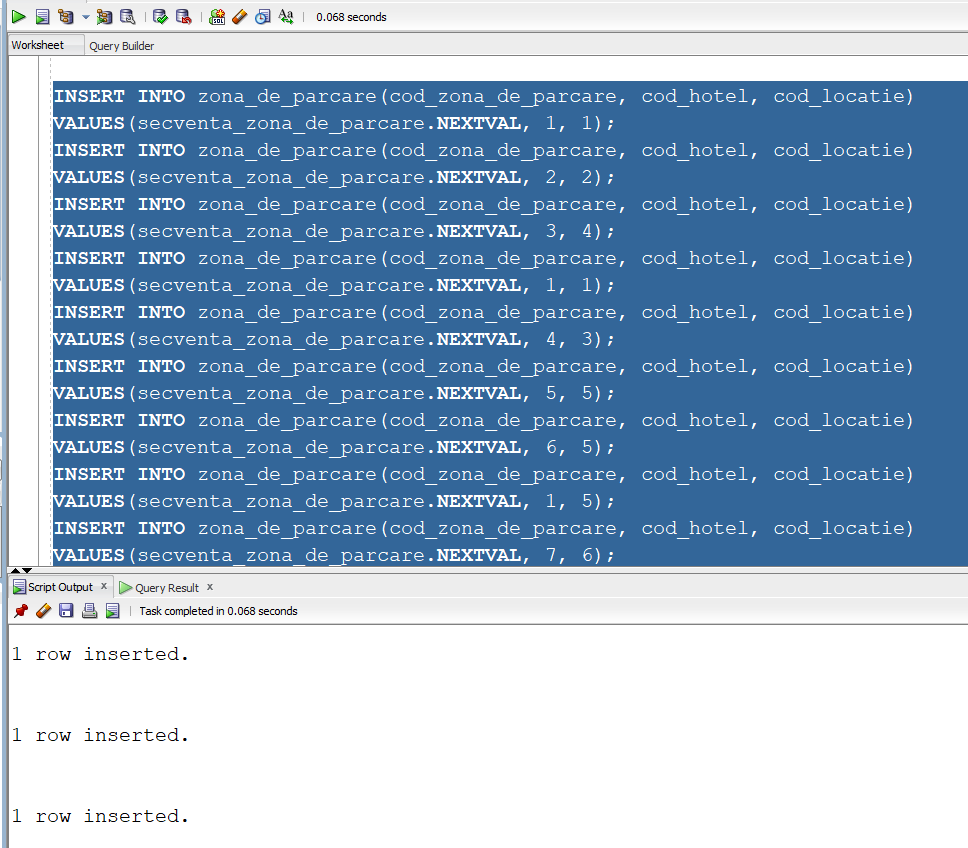


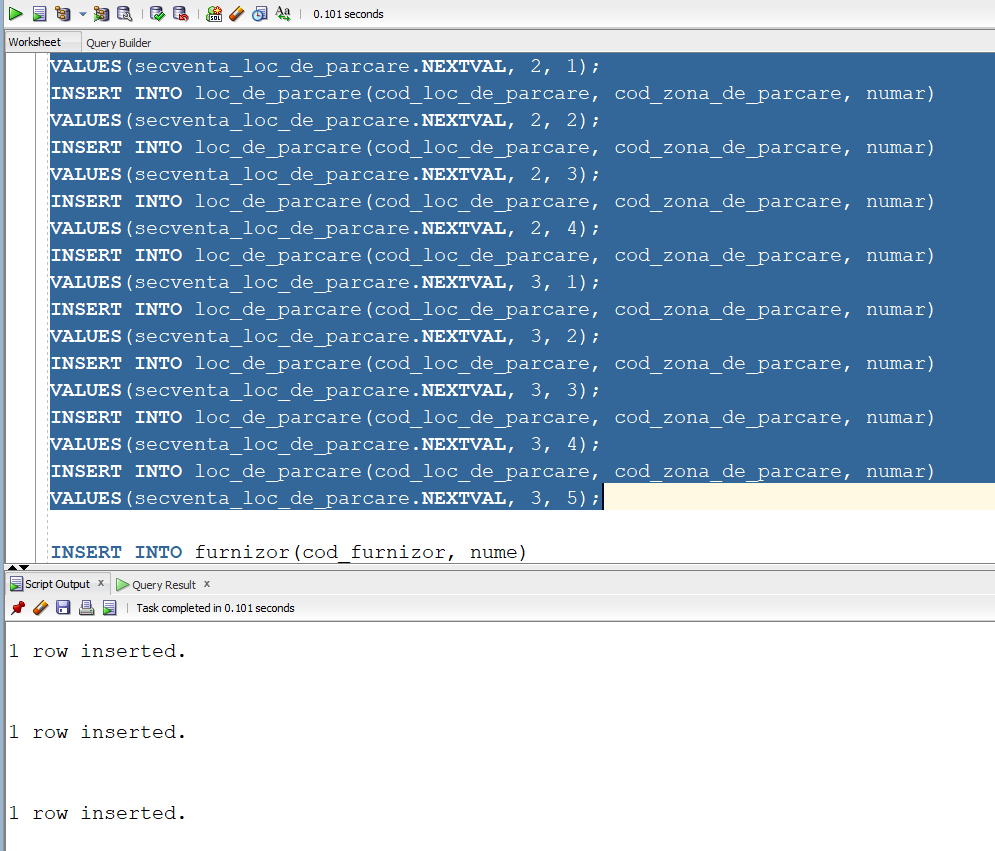


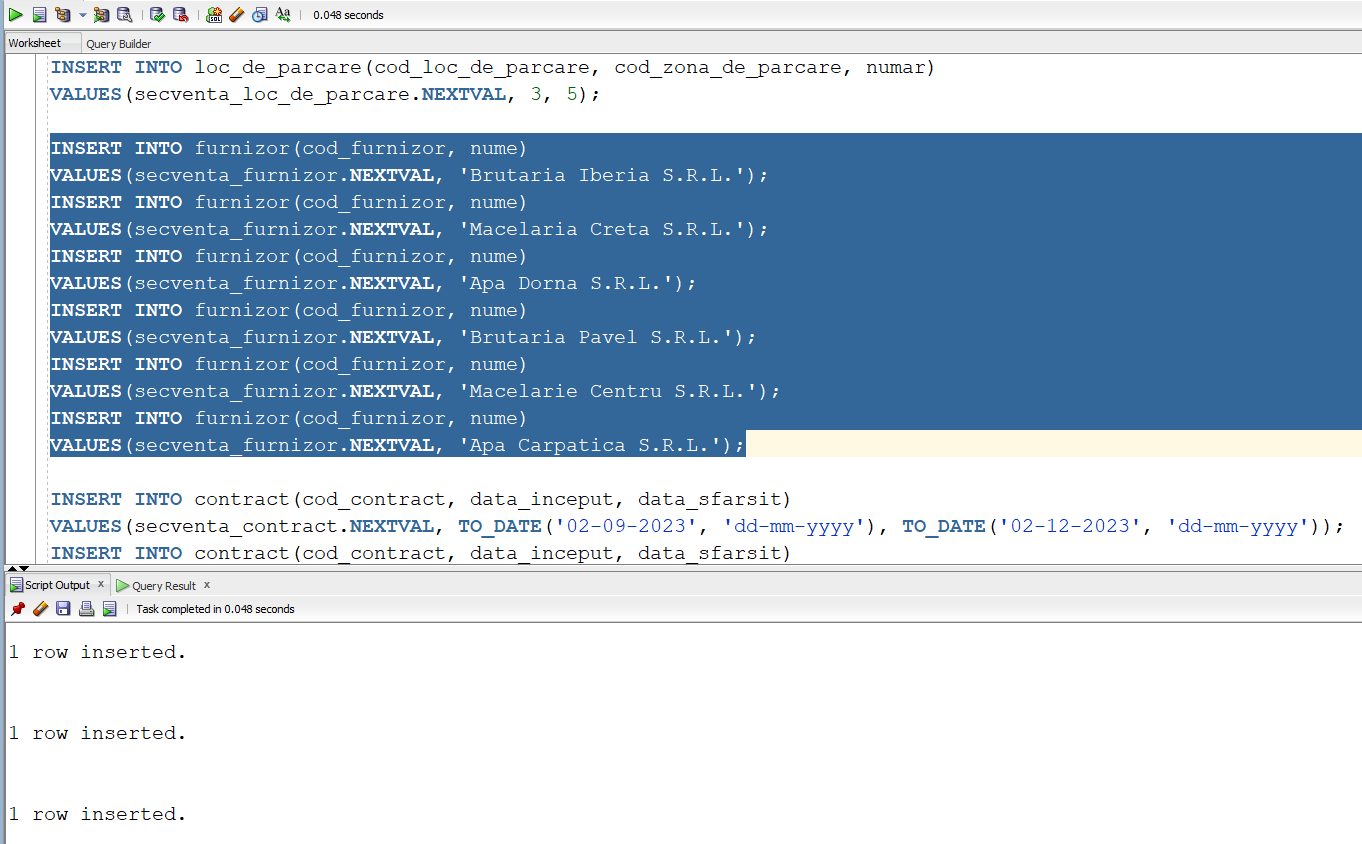


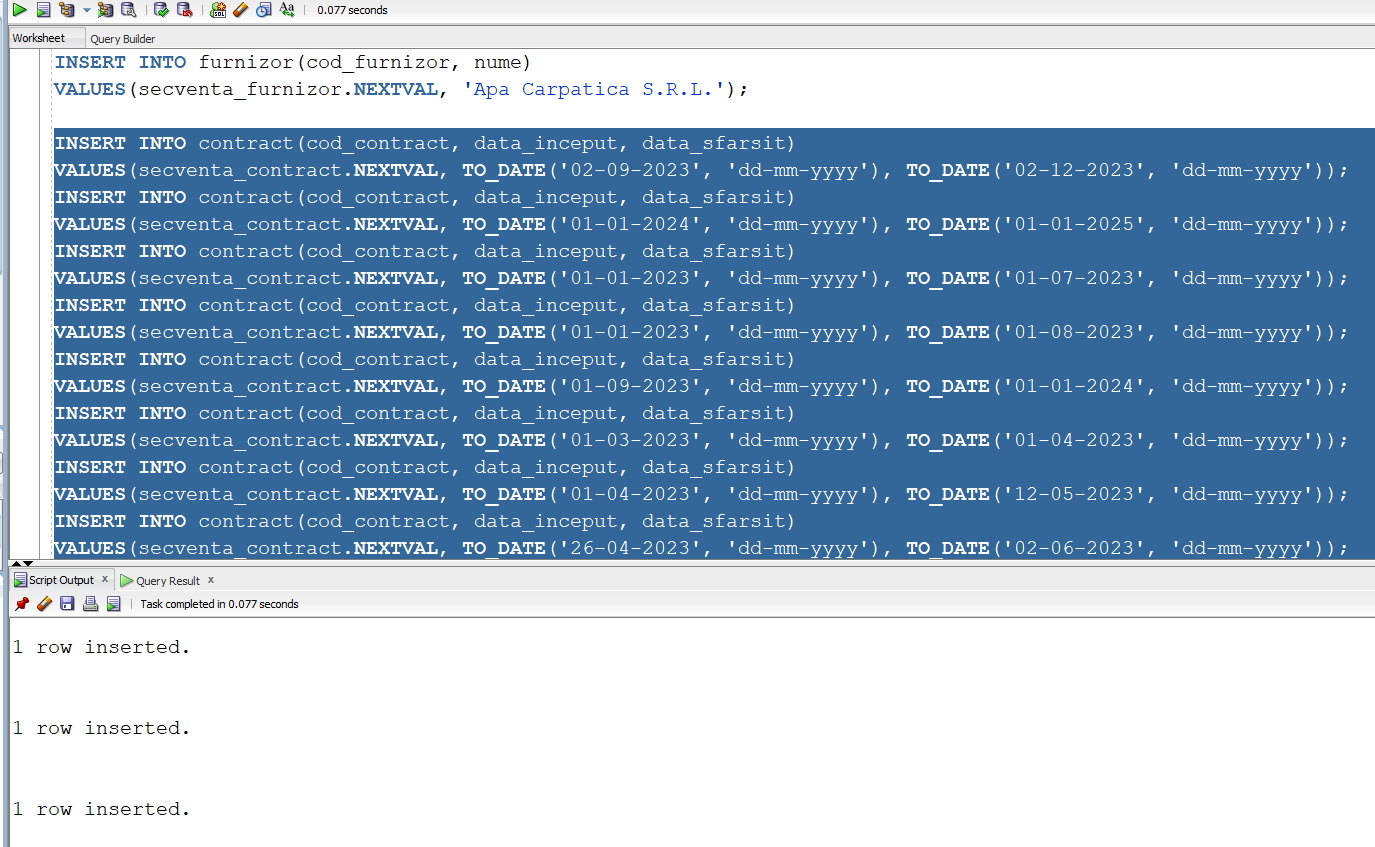


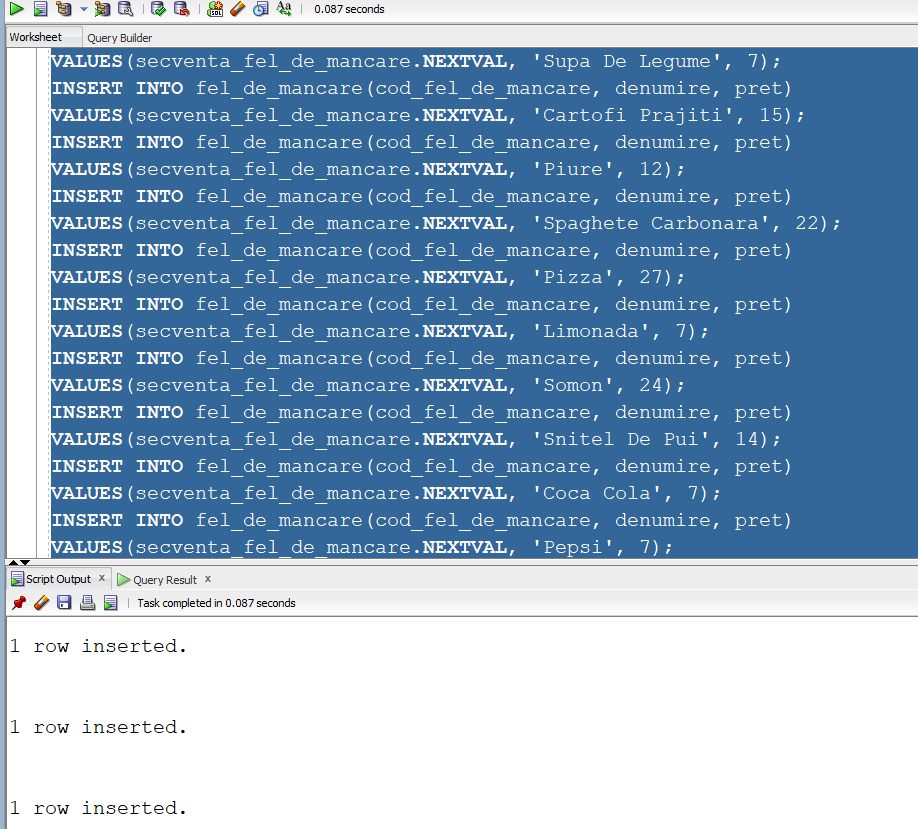


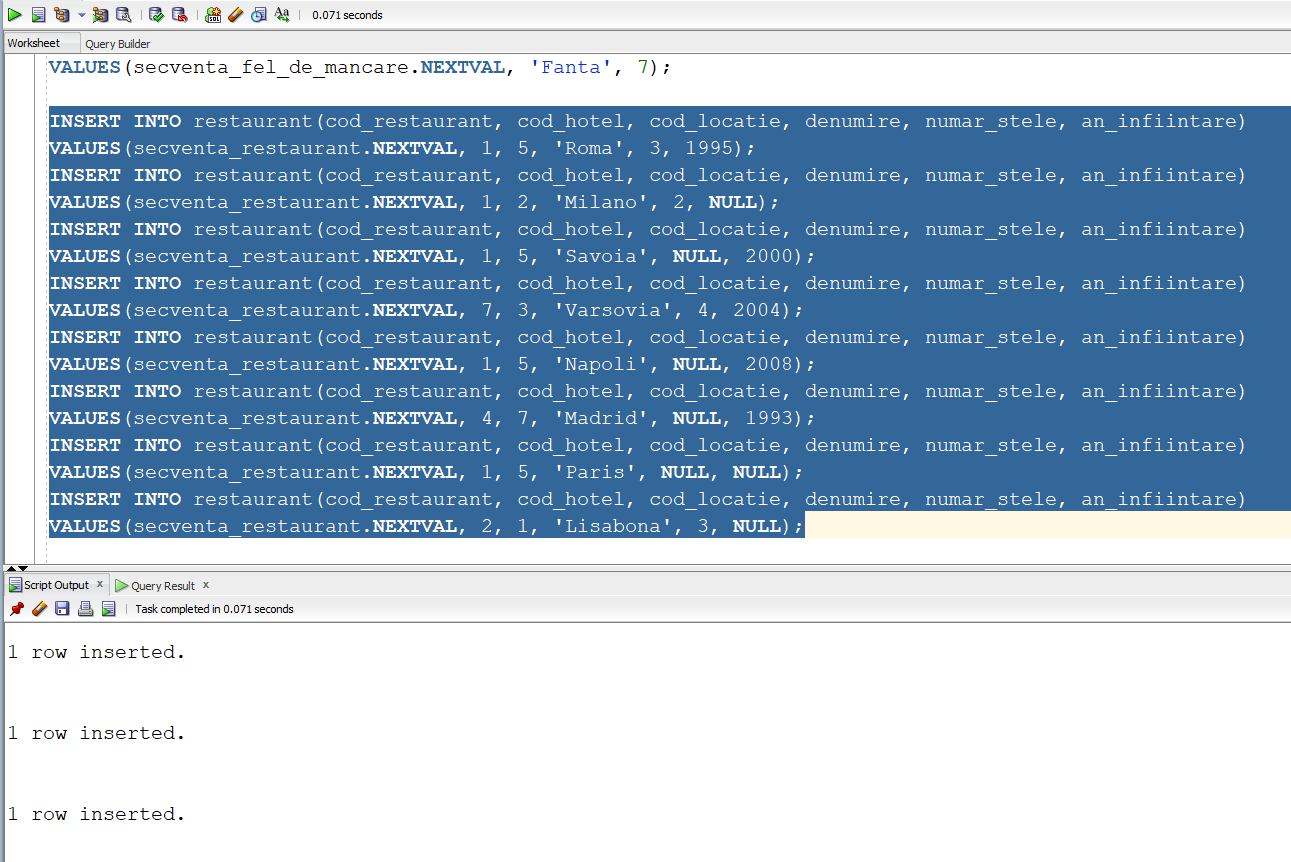


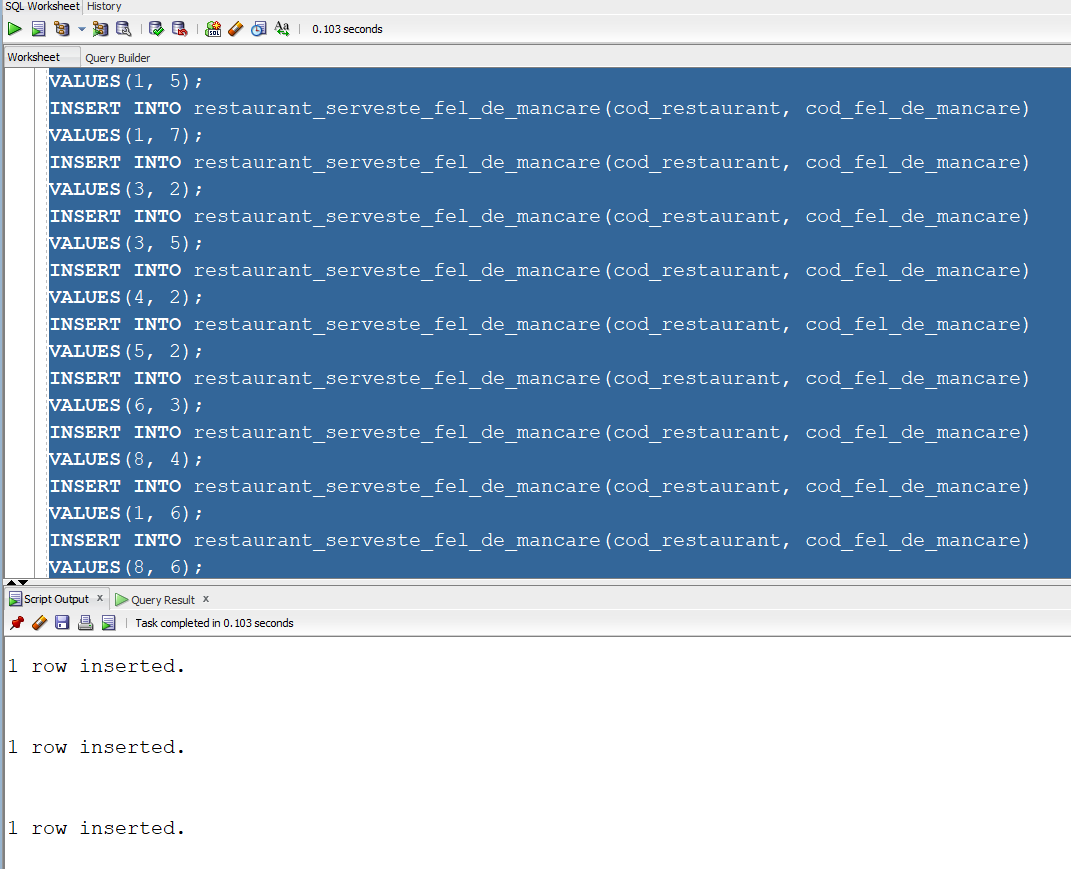


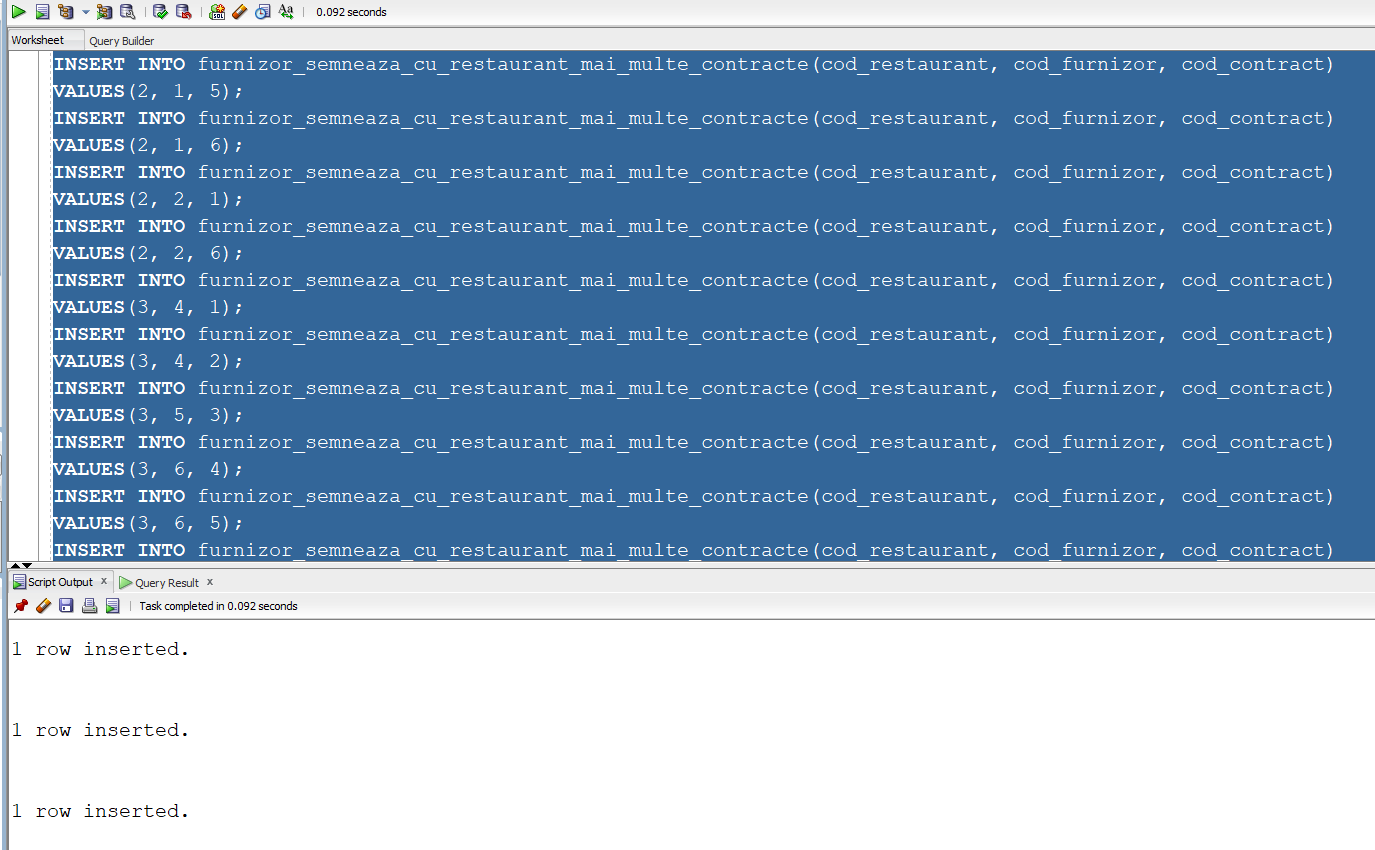


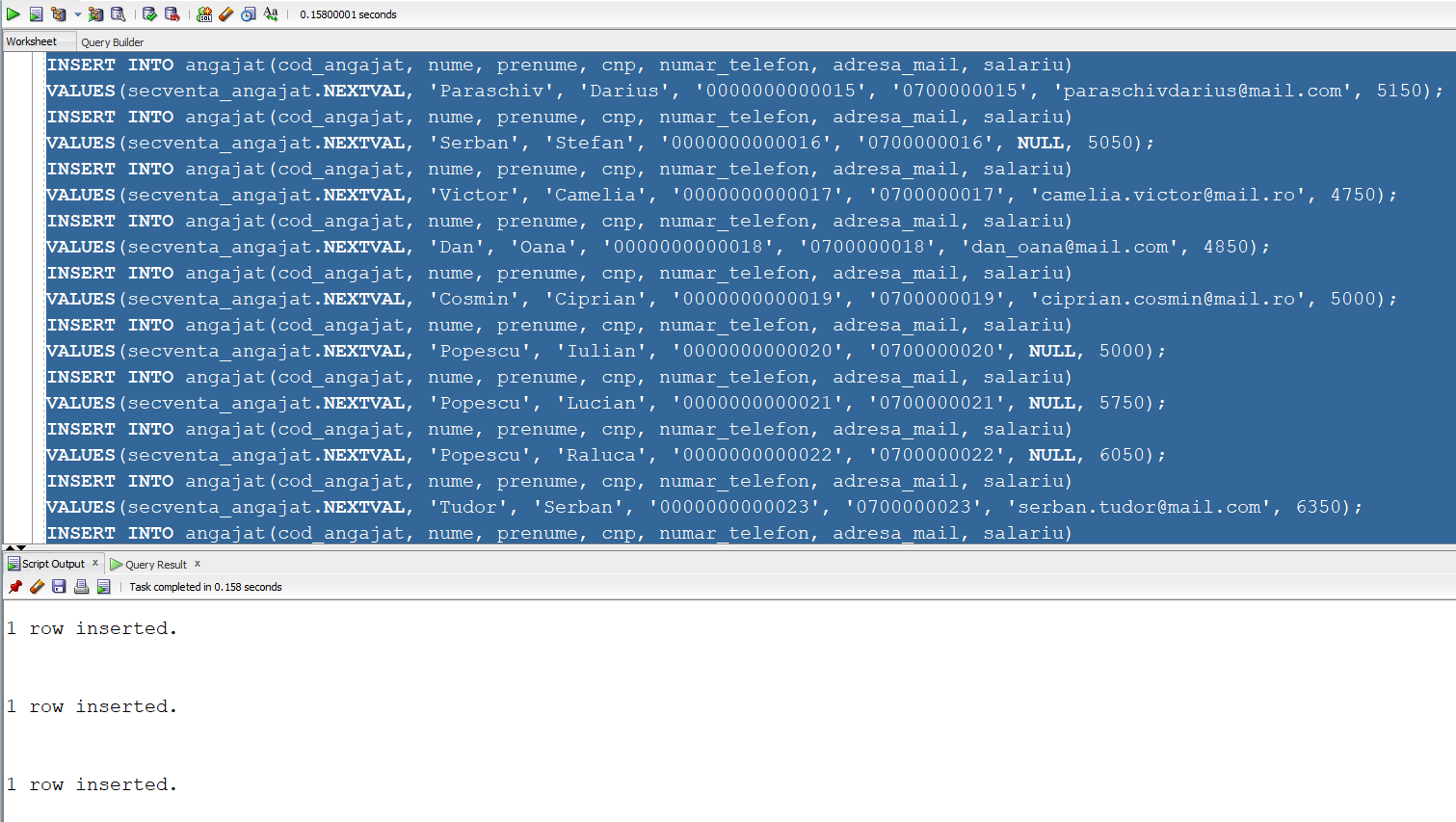


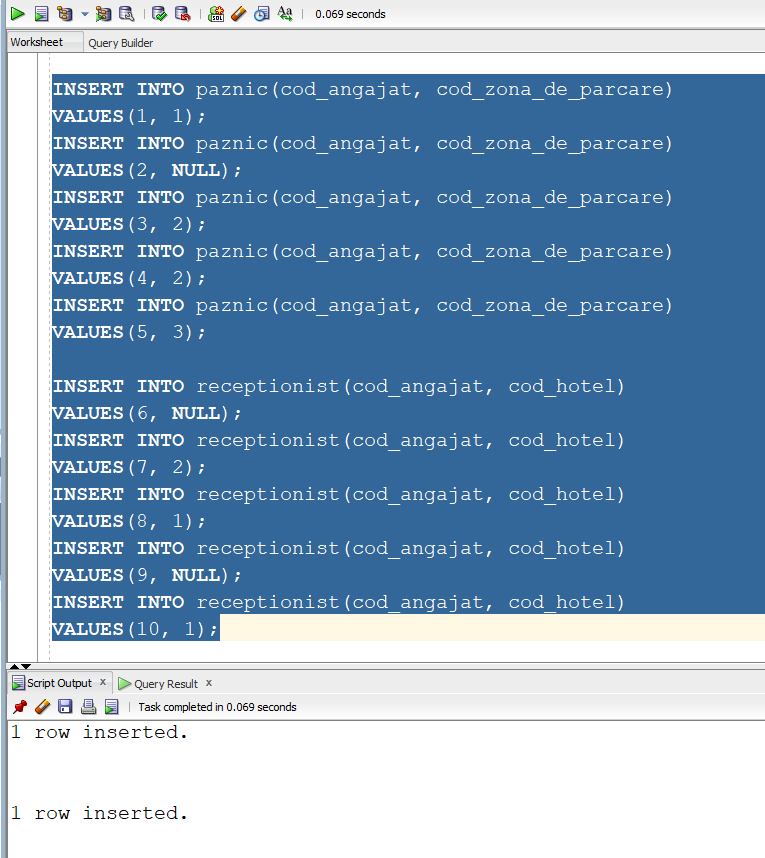


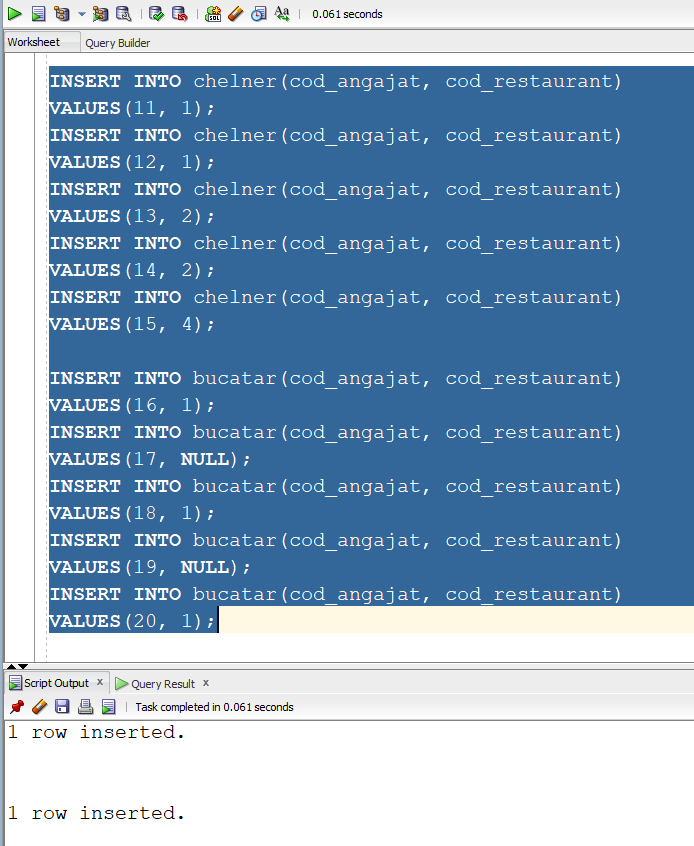


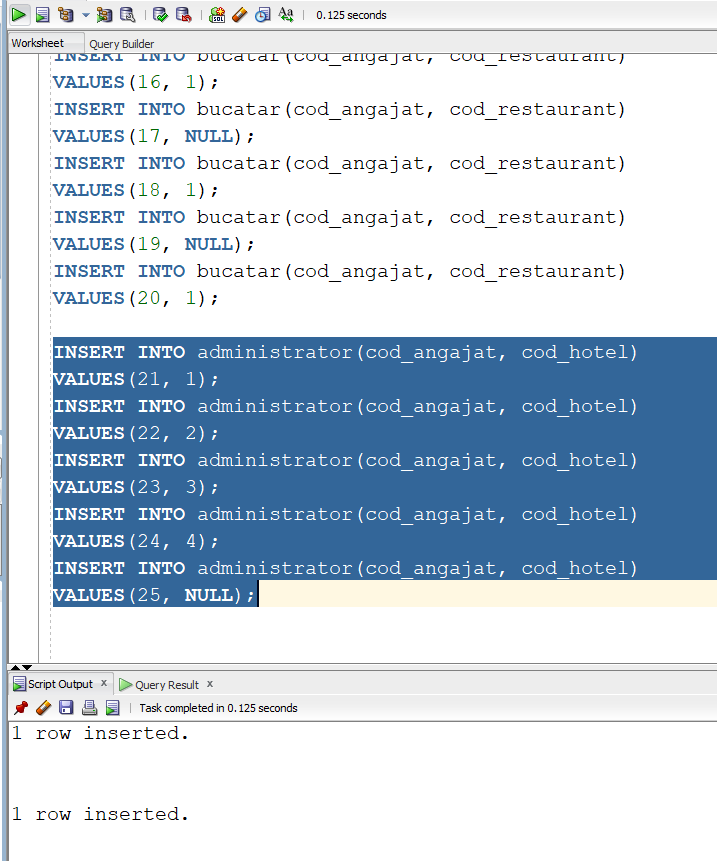


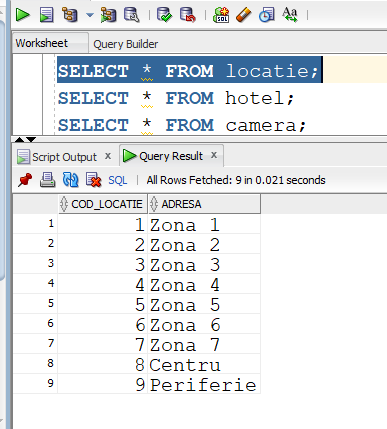


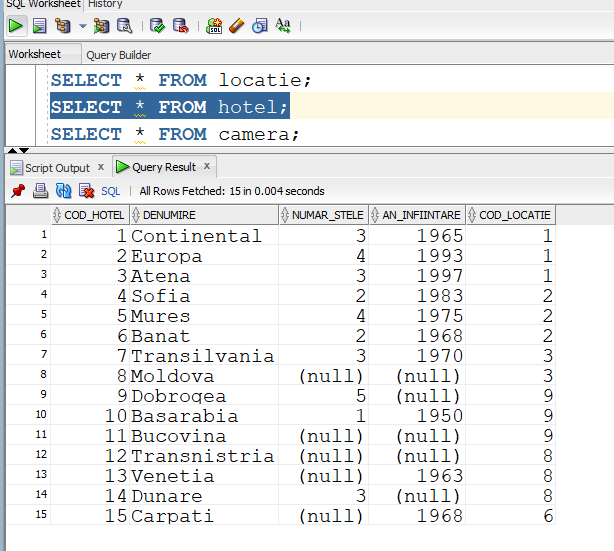


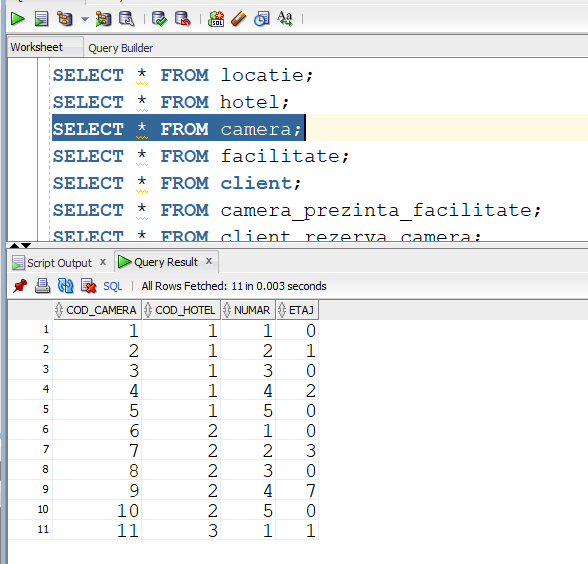


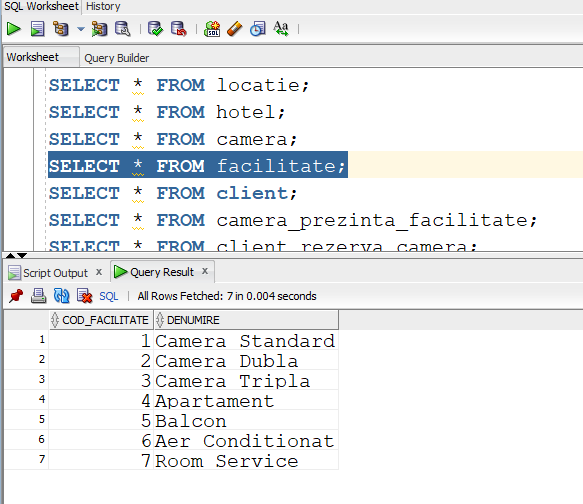


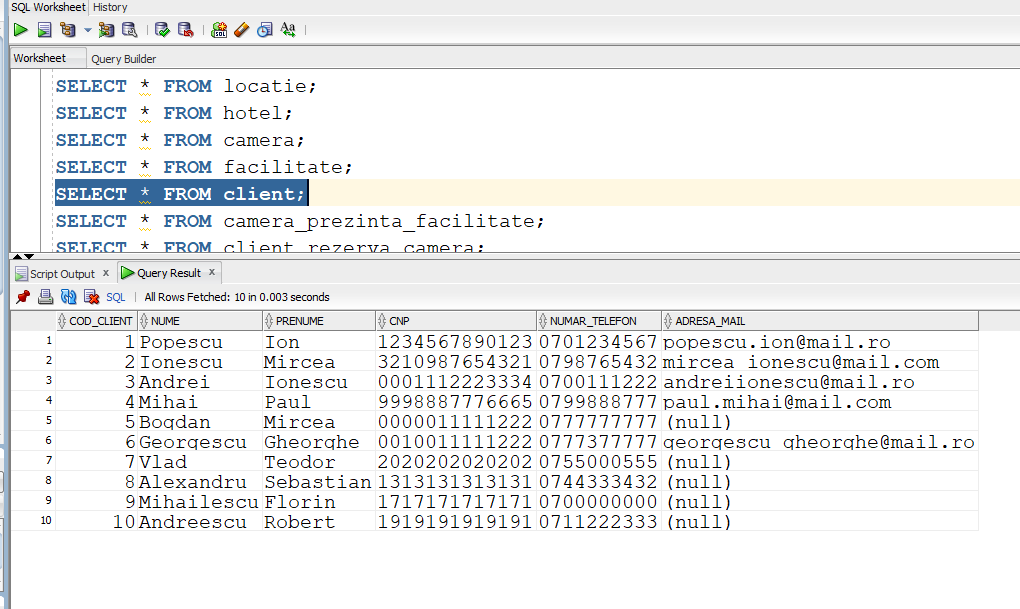


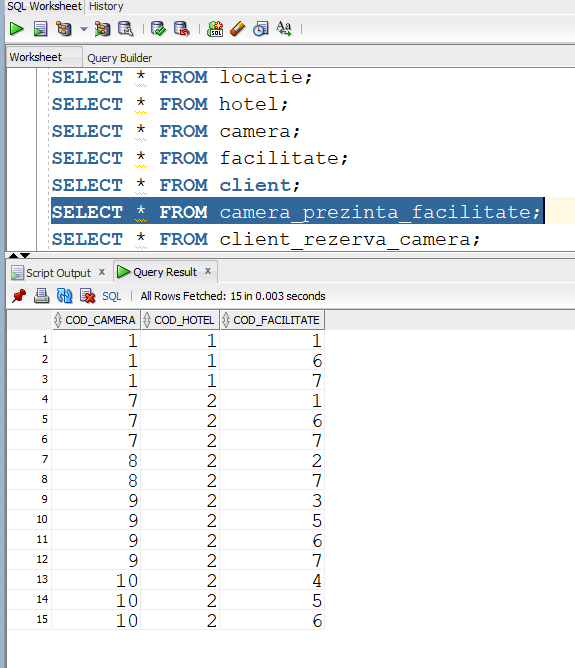


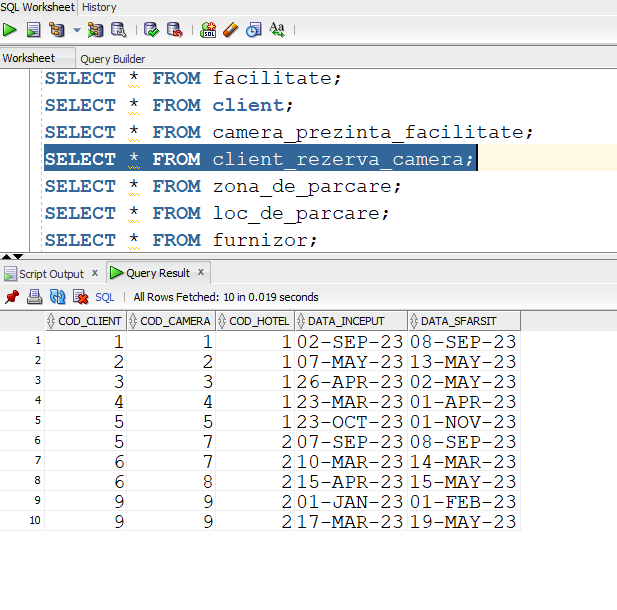


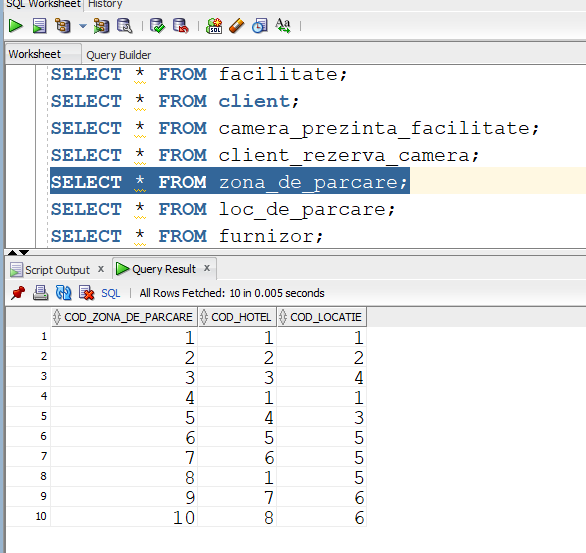


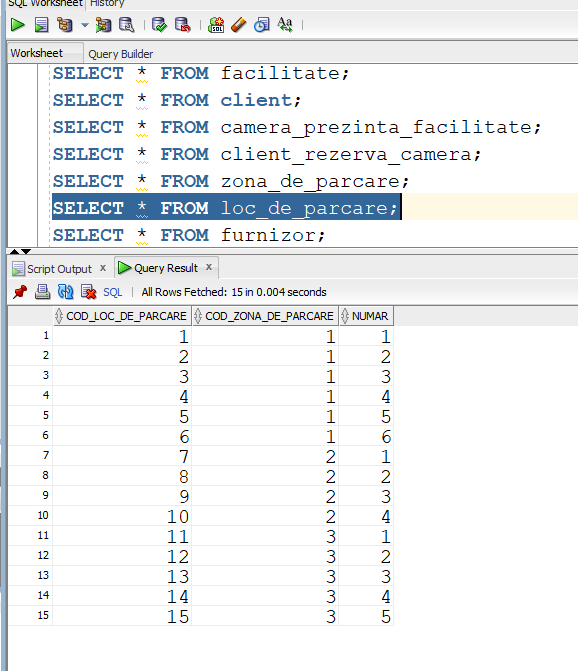


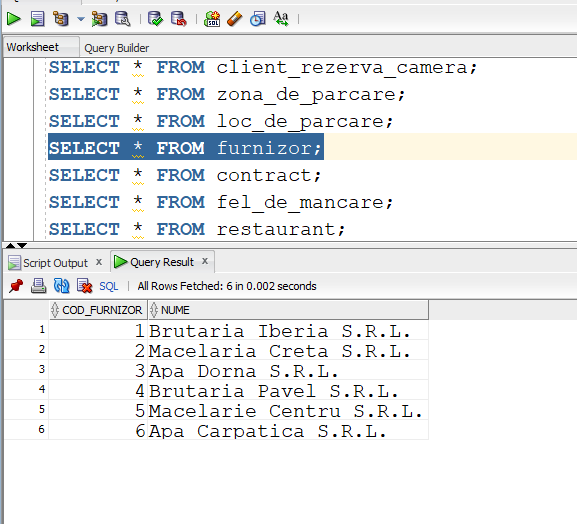


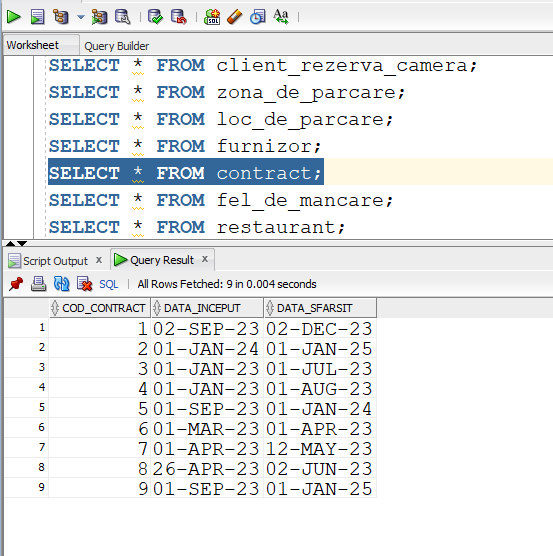


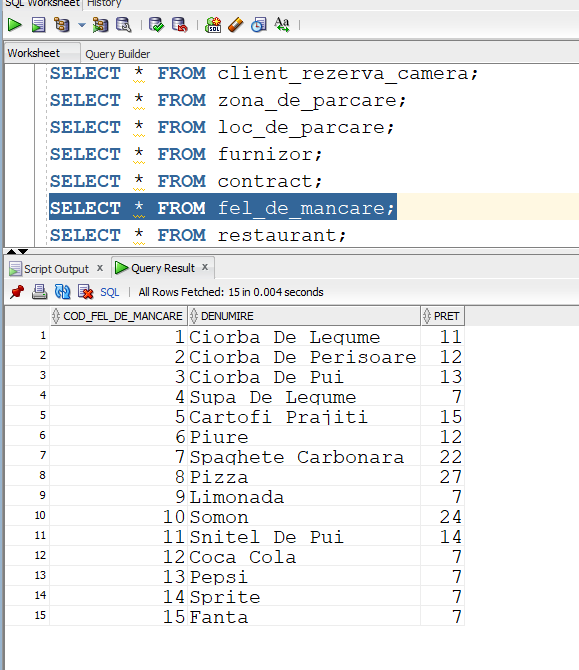


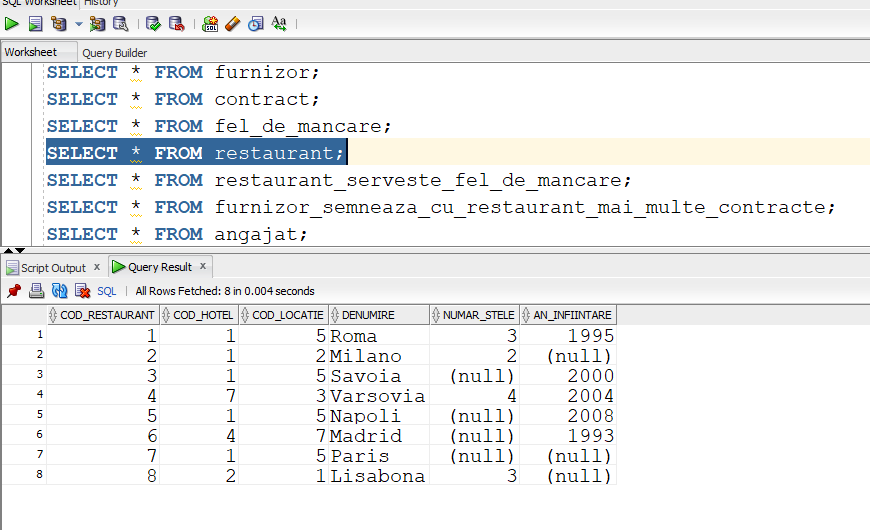


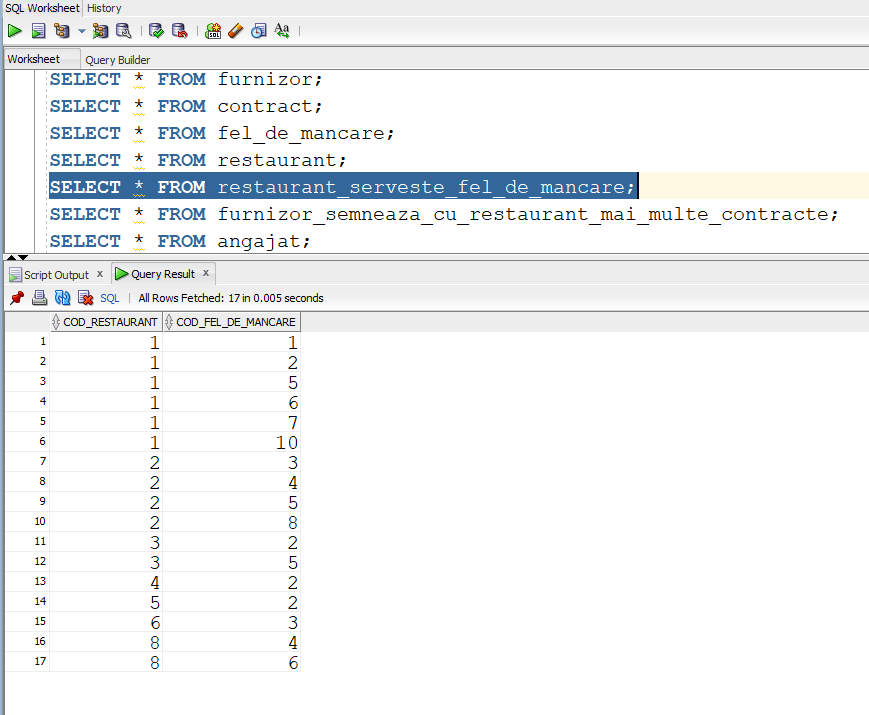


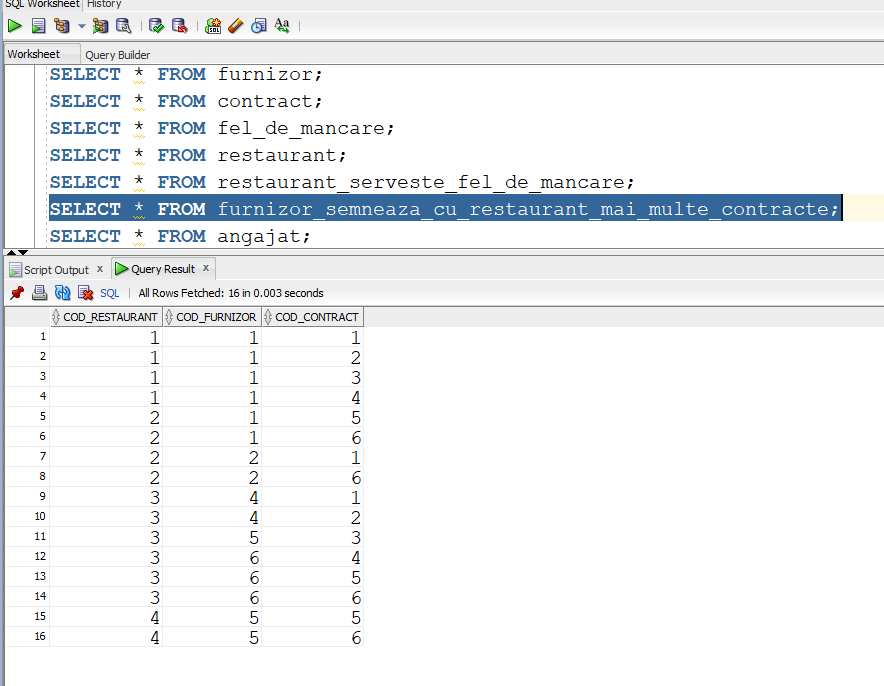


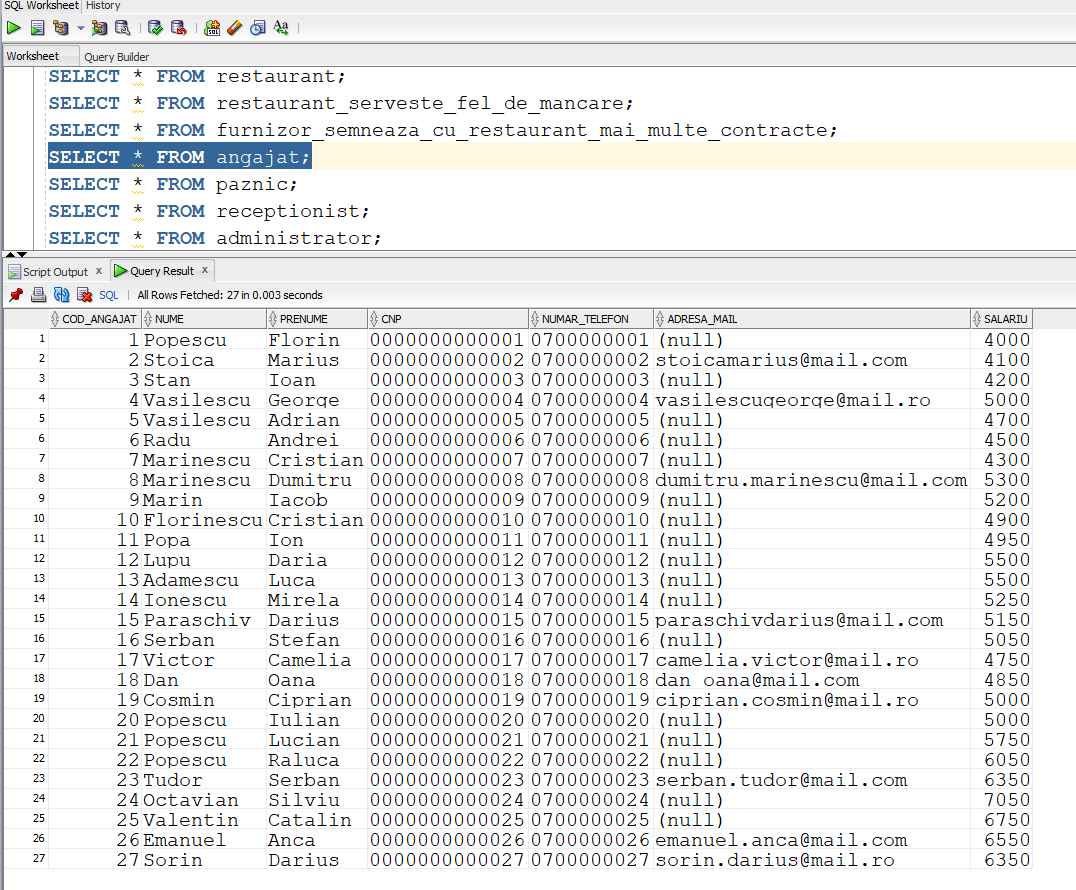


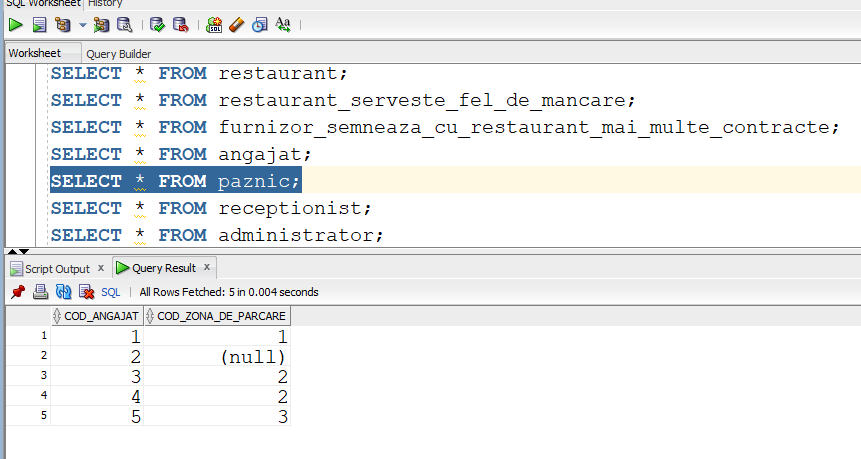


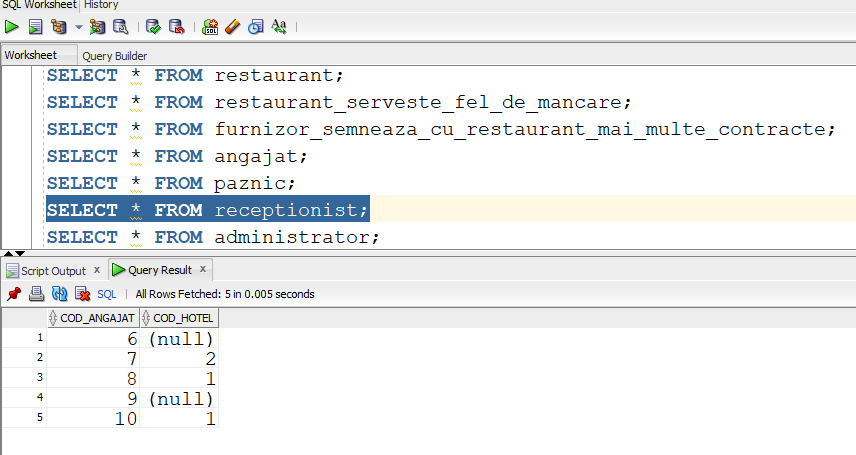


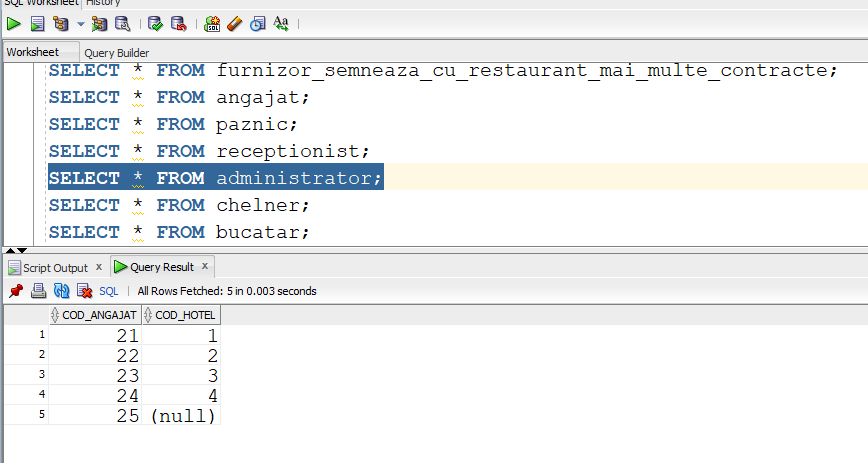


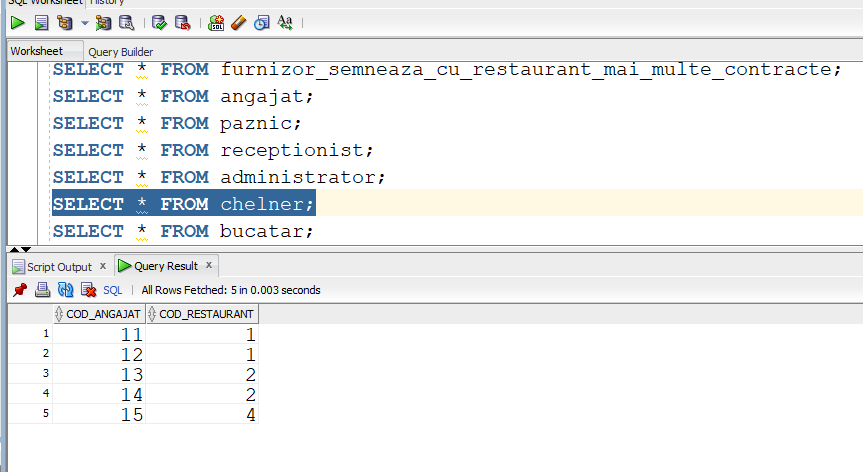


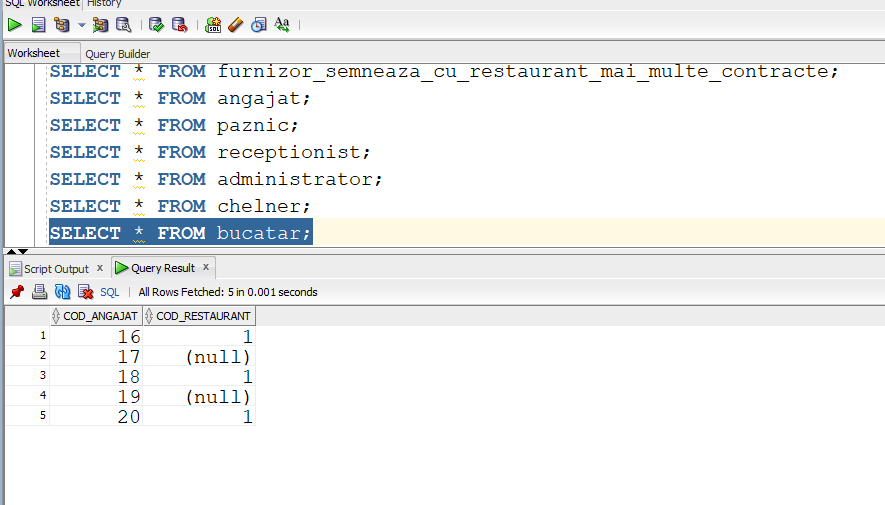












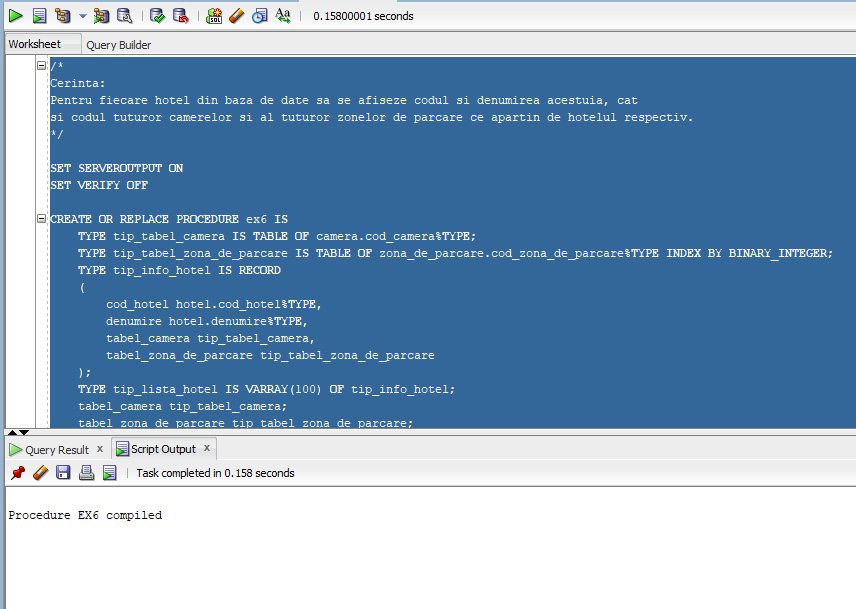
# Exercitiul 6

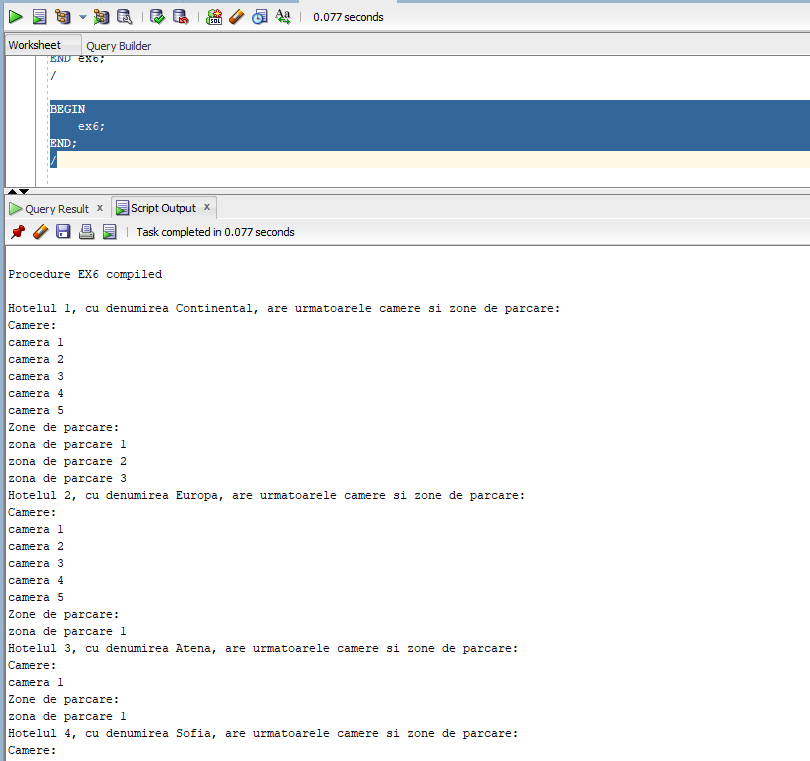
Formulati in limbaj natural o problema pe care sa o rezolvati folosind un subprogram stocat independent care sa utilizeze toate cele 3 tipuri de colectii studiate. Apelati subprogramul.

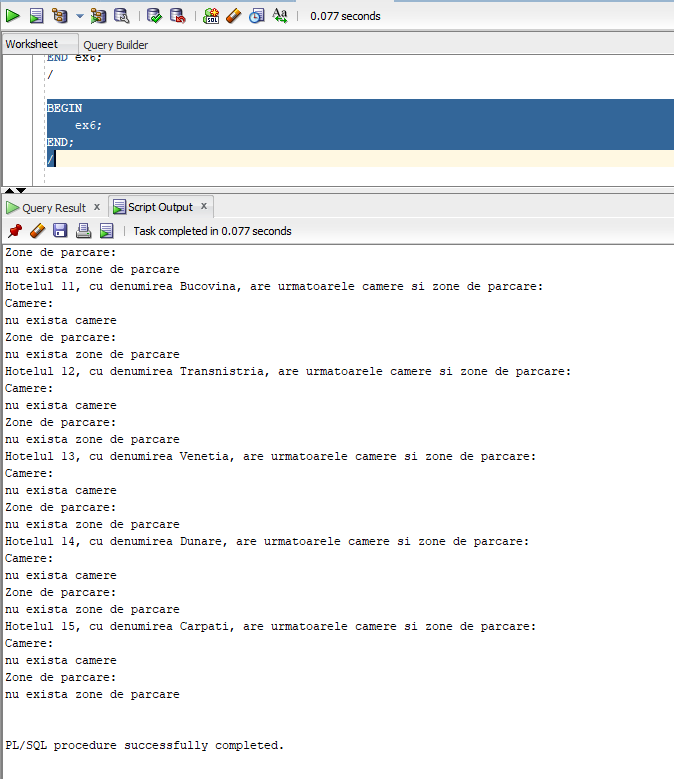
Cerinta:

Pentru fiecare hotel din baza de date sa se afiseze codul si denumirea acestuia, cat si codul tuturor camerelor si al tuturor zonelor de parcare ce apartin de hotelul respectiv.

|  |
| --- |
| /\*  Cerinta:  Pentru fiecare hotel din baza de date sa se afiseze codul si denumirea acestuia, cat  si codul tuturor camerelor si al tuturor zonelor de parcare ce apartin de hotelul respectiv.  \*/    SET SERVEROUTPUT ON  SET VERIFY OFF    CREATE OR REPLACE PROCEDURE ex6 IS  TYPE tip\_tabel\_camera IS TABLE OF camera.cod\_camera%TYPE;  TYPE tip\_tabel\_zona\_de\_parcare IS TABLE OF zona\_de\_parcare.cod\_zona\_de\_parcare%TYPE INDEX BY BINARY\_INTEGER;  TYPE tip\_info\_hotel IS RECORD  (  cod\_hotel hotel.cod\_hotel%TYPE,  denumire hotel.denumire%TYPE,  tabel\_camera tip\_tabel\_camera,  tabel\_zona\_de\_parcare tip\_tabel\_zona\_de\_parcare  );  TYPE tip\_lista\_hotel IS VARRAY(100) OF tip\_info\_hotel;  tabel\_camera tip\_tabel\_camera;  tabel\_zona\_de\_parcare tip\_tabel\_zona\_de\_parcare;  lista\_hotel tip\_lista\_hotel := tip\_lista\_hotel();  TYPE tip\_contor IS RECORD  (  cod\_hotel hotel.cod\_hotel%TYPE,  denumire hotel.denumire%TYPE  );  v\_contor tip\_contor;  BEGIN  lista\_hotel.DELETE;  FOR v\_contor IN (SELECT cod\_hotel, denumire FROM hotel)  LOOP  SELECT cod\_camera BULK COLLECT INTO tabel\_camera  FROM camera WHERE camera.cod\_hotel = v\_contor.cod\_hotel;    SELECT cod\_zona\_de\_parcare BULK COLLECT INTO tabel\_zona\_de\_parcare  FROM zona\_de\_parcare WHERE zona\_de\_parcare.cod\_hotel = v\_contor.cod\_hotel;    lista\_hotel.EXTEND;  lista\_hotel(lista\_hotel.LAST) := tip\_info\_hotel(v\_contor.cod\_hotel, v\_contor.denumire, tabel\_camera, tabel\_zona\_de\_parcare);  END LOOP;    FOR i IN lista\_hotel.FIRST..lista\_hotel.LAST  LOOP  DBMS\_OUTPUT.PUT\_LINE('Hotelul ' || lista\_hotel(i).cod\_hotel || ', cu denumirea ' || lista\_hotel(i).denumire || ', are urmatoarele camere si zone de parcare:');    DBMS\_OUTPUT.PUT\_LINE('Camere:');  IF lista\_hotel(i).tabel\_camera.COUNT = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('nu exista camere');  ELSE  FOR j IN lista\_hotel(i).tabel\_camera.FIRST..lista\_hotel(i).tabel\_camera.LAST  LOOP  DBMS\_OUTPUT.PUT\_LINE('camera ' || j);  END LOOP;  END IF;    DBMS\_OUTPUT.PUT\_LINE('Zone de parcare:');  IF lista\_hotel(i).tabel\_zona\_de\_parcare.COUNT = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('nu exista zone de parcare');  ELSE  FOR j IN lista\_hotel(i).tabel\_zona\_de\_parcare.FIRST..lista\_hotel(i).tabel\_zona\_de\_parcare.LAST  LOOP  DBMS\_OUTPUT.PUT\_LINE('zona de parcare ' || j);  END LOOP;  END IF;    END LOOP;  END ex6;  /    BEGIN  ex6;  END;  / |







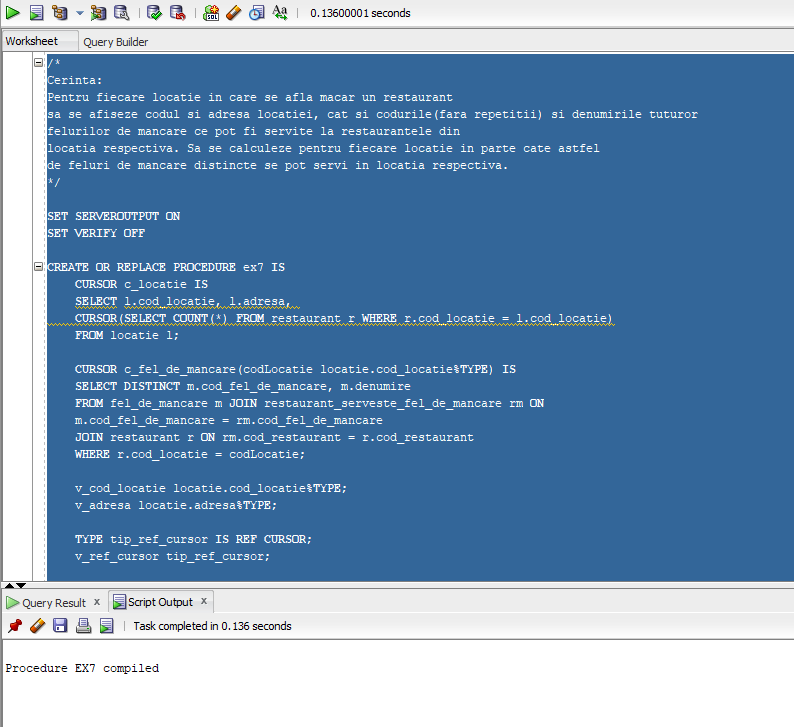
# Exercitiul 7

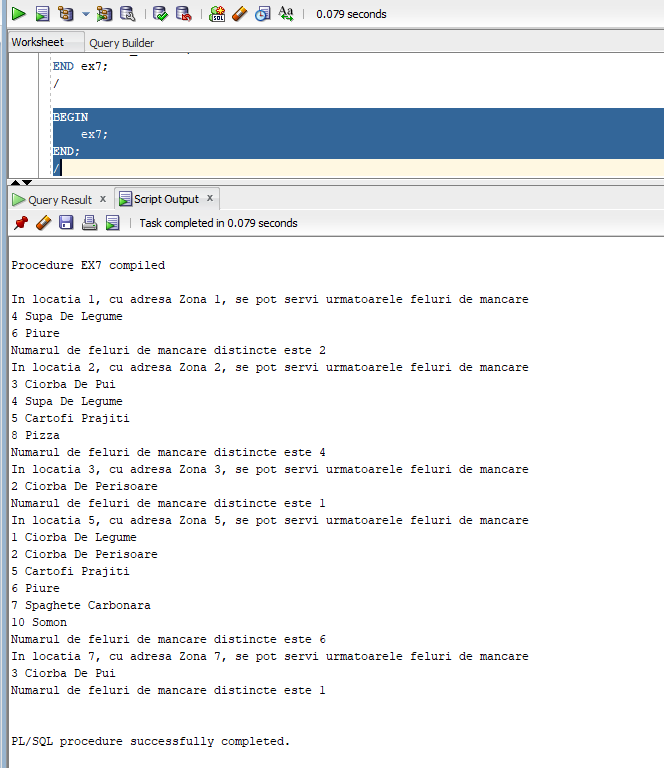
Formulati in limbaj natural o problema pe care sa o rezolvati folosind un subprogram stocat independent care sa utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celalalt cursor. Apelati subprogramul.

Cerinta:

Pentru fiecare locatie in care se afla macar un restaurant sa se afiseze codul si adresa locatiei, cat si codurile(fara repetitii) si denumirile tuturor felurilor de mancare ce pot fi servite la restaurantele din locatia respectiva. Sa se calculeze pentru fiecare locatie in parte cate astfel de feluri de mancare distincte se pot servi in locatia respectiva.

|  |
| --- |
| /\*  Cerinta:  Pentru fiecare locatie in care se afla macar un restaurant  sa se afiseze codul si adresa locatiei, cat si codurile(fara repetitii) si denumirile tuturor  felurilor de mancare ce pot fi servite la restaurantele din  locatia respectiva. Sa se calculeze pentru fiecare locatie in parte cate astfel  de feluri de mancare distincte se pot servi in locatia respectiva.  \*/    SET SERVEROUTPUT ON  SET VERIFY OFF    CREATE OR REPLACE PROCEDURE ex7 IS  CURSOR c\_locatie IS  SELECT l.cod\_locatie, l.adresa,  CURSOR(SELECT COUNT(\*) FROM restaurant r WHERE r.cod\_locatie = l.cod\_locatie)  FROM locatie l;    CURSOR c\_fel\_de\_mancare(codLocatie locatie.cod\_locatie%TYPE) IS  SELECT DISTINCT m.cod\_fel\_de\_mancare, m.denumire  FROM fel\_de\_mancare m JOIN restaurant\_serveste\_fel\_de\_mancare rm ON  m.cod\_fel\_de\_mancare = rm.cod\_fel\_de\_mancare  JOIN restaurant r ON rm.cod\_restaurant = r.cod\_restaurant  WHERE r.cod\_locatie = codLocatie;    v\_cod\_locatie locatie.cod\_locatie%TYPE;  v\_adresa locatie.adresa%TYPE;    TYPE tip\_ref\_cursor IS REF CURSOR;  v\_ref\_cursor tip\_ref\_cursor;    v\_numar\_restaurante NUMBER(10);    v\_cod\_fel\_de\_mancare fel\_de\_mancare.cod\_fel\_de\_mancare%TYPE;  v\_denumire fel\_de\_mancare.denumire%TYPE;    v\_numar\_feluri\_de\_mancare NUMBER(10);  BEGIN  OPEN c\_locatie;  LOOP  FETCH c\_locatie INTO v\_cod\_locatie, v\_adresa, v\_ref\_cursor;    EXIT WHEN c\_locatie%NOTFOUND;    v\_numar\_restaurante := 0;    LOOP  FETCH v\_ref\_cursor INTO v\_numar\_restaurante;    EXIT WHEN v\_ref\_cursor%NOTFOUND;  END LOOP;    IF v\_numar\_restaurante > 0 THEN  DBMS\_OUTPUT.PUT\_LINE('In locatia ' || v\_cod\_locatie || ', cu adresa ' || v\_adresa || ', se pot servi urmatoarele feluri de mancare');    v\_numar\_feluri\_de\_mancare := 0;    OPEN c\_fel\_de\_mancare(v\_cod\_locatie);  LOOP  FETCH c\_fel\_de\_mancare INTO v\_cod\_fel\_de\_mancare, v\_denumire;  EXIT WHEN c\_fel\_de\_mancare%NOTFOUND;    v\_numar\_feluri\_de\_mancare := v\_numar\_feluri\_de\_mancare + 1;    DBMS\_OUTPUT.PUT\_LINE(v\_cod\_fel\_de\_mancare || ' ' || v\_denumire);  END LOOP;  CLOSE c\_fel\_de\_mancare;    DBMS\_OUTPUT.PUT\_LINE('Numarul de feluri de mancare distincte este ' || v\_numar\_feluri\_de\_mancare);    END IF;    END LOOP;  CLOSE c\_locatie;  END ex7;  /    BEGIN  ex7;  END;  / |





# Exercitiul 8

Formulati in limbaj natural o problema pe care sa o rezolvati folosind un subprogram stocat independent de tip functie care sa utilizeze intr-o singura comanda SQL 3 dintre tabelele definite. Definiti minim 2 exceptii proprii. Apelati subprogramul astfel incat sa evidentiati toate cazurile definite si tratate.

Cerinta:

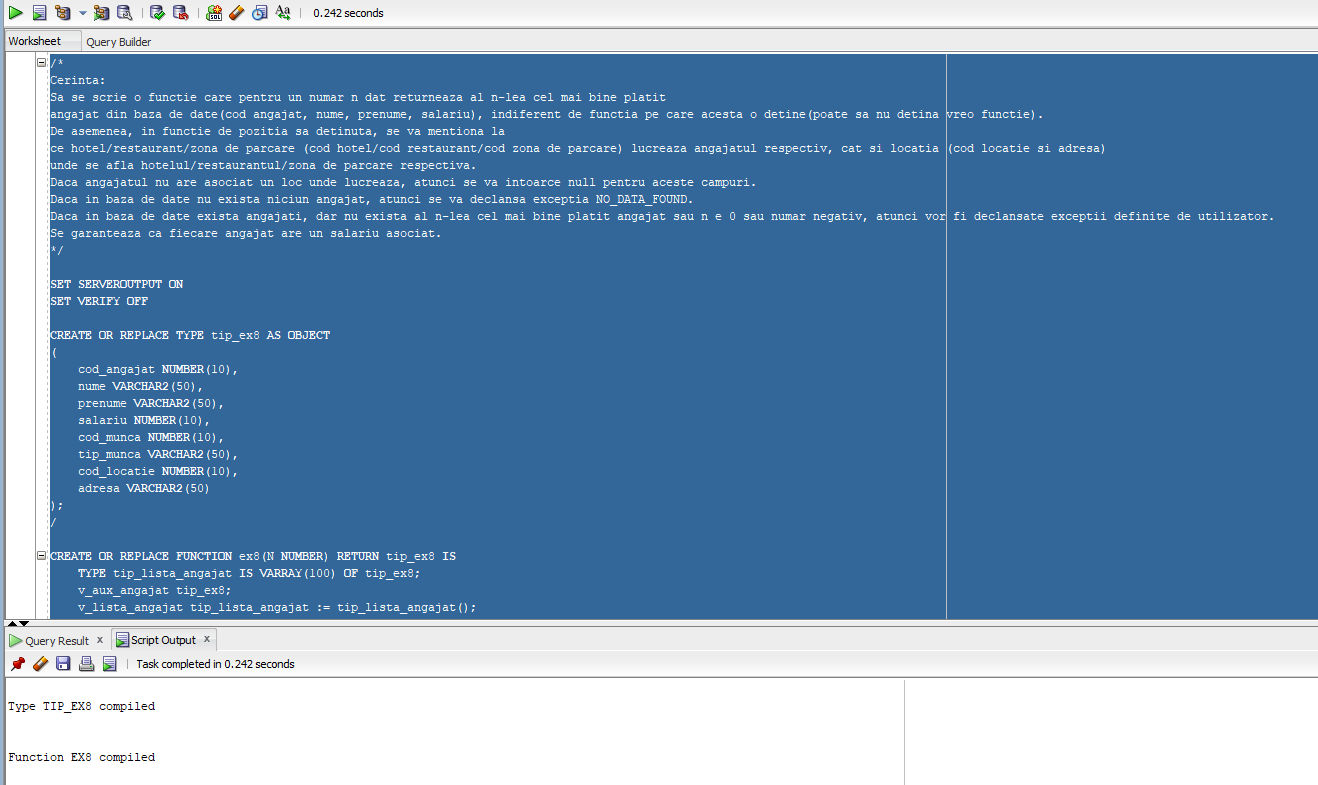
Sa se scrie o functie care pentru un numar n dat returneaza al n-lea cel mai bine platit angajat din baza de date(cod angajat, nume, prenume, salariu), indiferent de functia pe care acesta o detine(poate sa nu detina vreo functie). De asemenea, in functie de pozitia sa detinuta, se va mentiona la ce hotel/restaurant/zona de parcare (cod hotel/cod restaurant/cod zona de parcare) lucreaza angajatul respectiv, cat si locatia (cod locatie si adresa) unde se afla hotelul/restaurantul/zona de parcare respectiva.

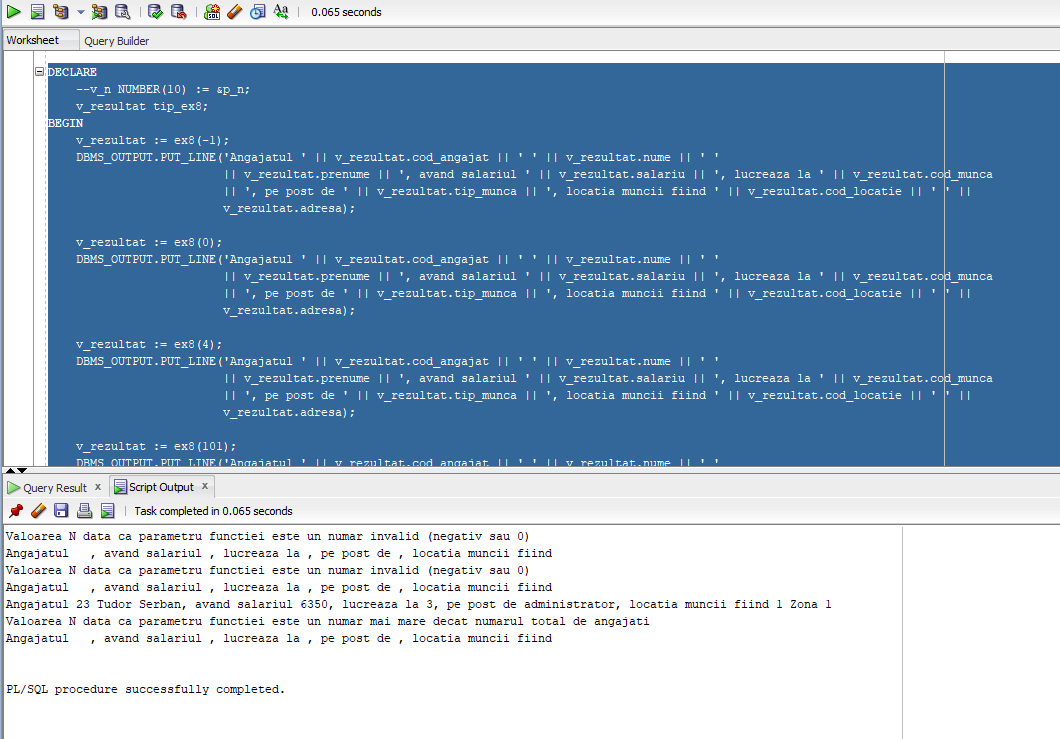
Daca angajatul nu are asociat un loc unde lucreaza, atunci se va intoarce null pentru aceste campuri. Daca in baza de date nu exista niciun angajat, atunci se va declansa exceptia NO\_DATA\_FOUND.

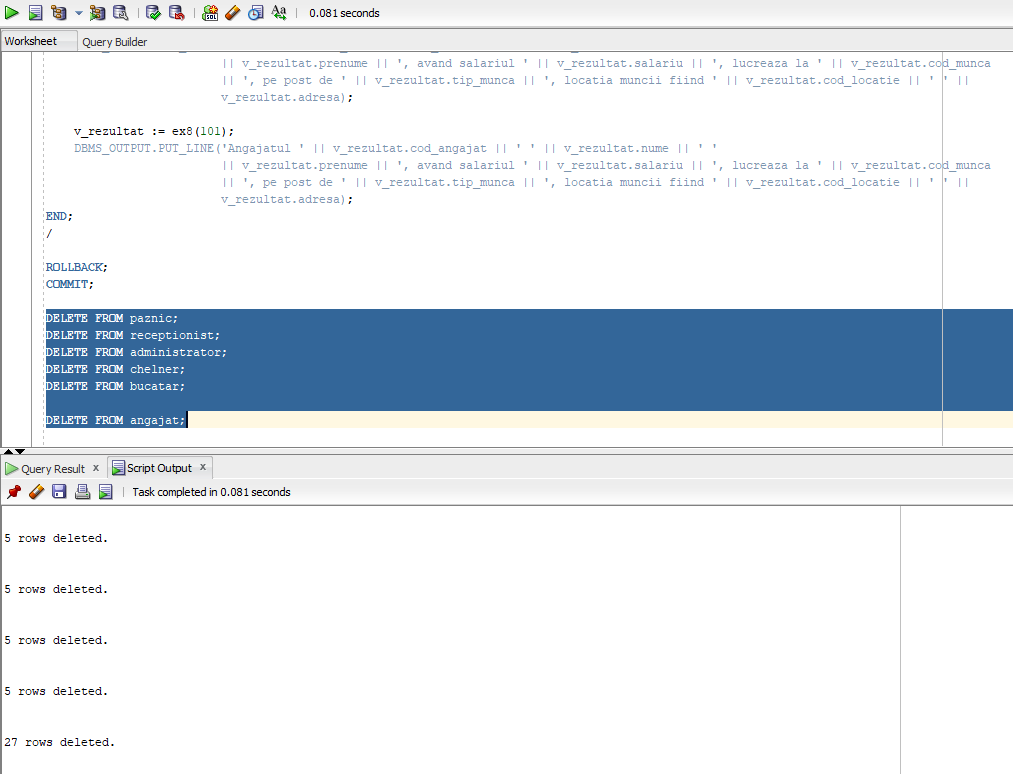
Daca in baza de date exista angajati, dar nu exista al n-lea cel mai bine platit angajat sau n e 0 sau numar negativ, atunci vor fi declansate exceptii definite de utilizator.

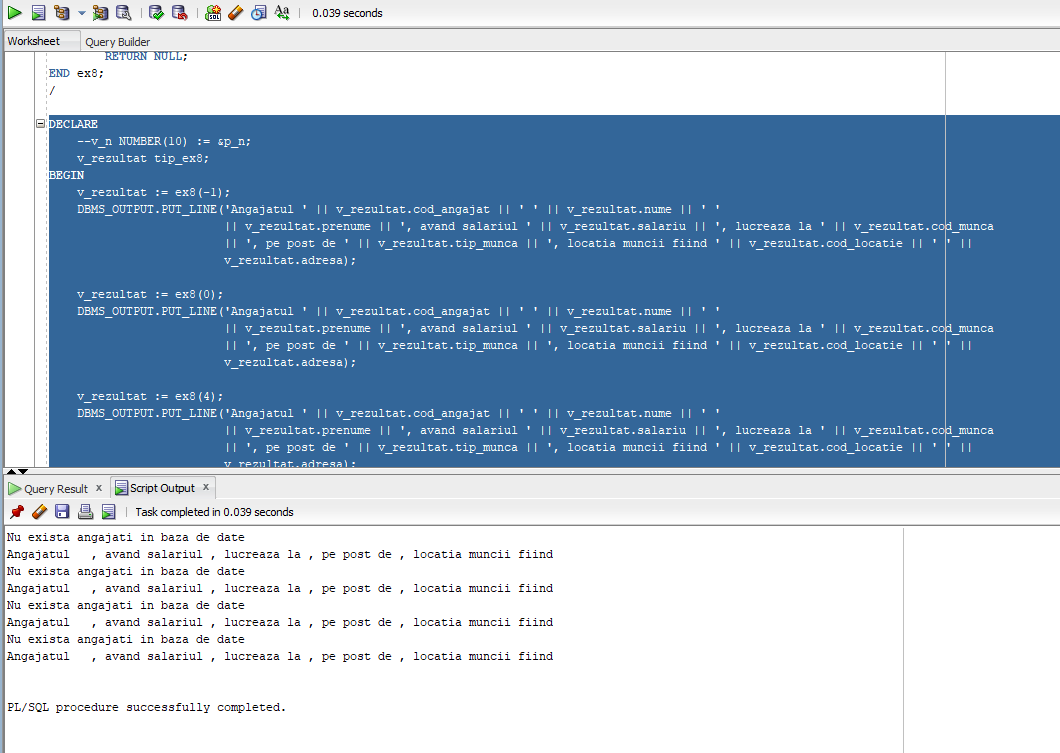
Se garanteaza ca fiecare angajat are un salariu asociat.

|  |
| --- |
| /\*  Cerinta:  Sa se scrie o functie care pentru un numar n dat returneaza al n-lea cel mai bine platit  angajat din baza de date(cod angajat, nume, prenume, salariu), indiferent de functia pe care acesta o detine(poate sa nu detina vreo functie).  De asemenea, in functie de pozitia sa detinuta, se va mentiona la  ce hotel/restaurant/zona de parcare (cod hotel/cod restaurant/cod zona de parcare) lucreaza angajatul respectiv, cat si locatia (cod locatie si adresa)  unde se afla hotelul/restaurantul/zona de parcare respectiva.  Daca angajatul nu are asociat un loc unde lucreaza, atunci se va intoarce null pentru aceste campuri.  Daca in baza de date nu exista niciun angajat, atunci se va declansa exceptia NO\_DATA\_FOUND.  Daca in baza de date exista angajati, dar nu exista al n-lea cel mai bine platit angajat sau n e 0 sau numar negativ, atunci vor fi declansate exceptii definite de utilizator.  Se garanteaza ca fiecare angajat are un salariu asociat.  \*/    SET SERVEROUTPUT ON  SET VERIFY OFF    CREATE OR REPLACE TYPE tip\_ex8 AS OBJECT  (  cod\_angajat NUMBER(10),  nume VARCHAR2(50),  prenume VARCHAR2(50),  salariu NUMBER(10),  cod\_munca NUMBER(10),  tip\_munca VARCHAR2(50),  cod\_locatie NUMBER(10),  adresa VARCHAR2(50)  );  /    CREATE OR REPLACE FUNCTION ex8(N NUMBER) RETURN tip\_ex8 IS  TYPE tip\_lista\_angajat IS VARRAY(100) OF tip\_ex8;  v\_aux\_angajat tip\_ex8;  v\_lista\_angajat tip\_lista\_angajat := tip\_lista\_angajat();  v\_lista\_aux\_angajat tip\_lista\_angajat := tip\_lista\_angajat();    v\_sortat BOOLEAN;  v\_deja\_luat BOOLEAN;    N\_INVALID EXCEPTION;  N\_PREA\_MARE EXCEPTION;    PRAGMA EXCEPTION\_INIT(N\_INVALID, -20001);  PRAGMA EXCEPTION\_INIT(N\_PREA\_MARE, -20002);    v\_nr\_angajati NUMBER(10) := 0;    BEGIN  SELECT COUNT(\*) INTO v\_nr\_angajati  FROM angajat;    IF v\_nr\_angajati = 0 THEN  RAISE NO\_DATA\_FOUND;  END IF;    IF N <= 0 THEN  RAISE N\_INVALID;  END IF;    IF v\_nr\_angajati < N THEN  RAISE N\_PREA\_MARE;  END IF;    --paznic  SELECT tip\_ex8(  p.cod\_angajat,  a.nume,  a.prenume,  a.salariu,  p.cod\_zona\_de\_parcare,  'paznic',  l.cod\_locatie,  l.adresa  )  BULK COLLECT INTO v\_lista\_aux\_angajat  FROM paznic p LEFT JOIN angajat a ON (p.cod\_angajat = a.cod\_angajat)  LEFT JOIN zona\_de\_parcare z ON (p.cod\_zona\_de\_parcare = z.cod\_zona\_de\_parcare)  LEFT JOIN locatie l ON (z.cod\_locatie = l.cod\_locatie);    FOR i IN v\_lista\_aux\_angajat.FIRST..v\_lista\_aux\_angajat.LAST  LOOP  v\_deja\_luat := FALSE;    IF v\_lista\_angajat.COUNT > 0 THEN  FOR j IN v\_lista\_angajat.FIRST..v\_lista\_angajat.LAST  LOOP  IF v\_lista\_aux\_angajat(i).cod\_angajat = v\_lista\_angajat(j).cod\_angajat THEN  v\_deja\_luat := TRUE;  END IF;  END LOOP;  END IF;    IF v\_deja\_luat = FALSE THEN  v\_lista\_angajat.EXTEND;  v\_lista\_angajat(v\_lista\_angajat.COUNT) := v\_lista\_aux\_angajat(i);  END IF;  END LOOP;  --receptionist  SELECT tip\_ex8(  r.cod\_angajat,  a.nume,  a.prenume,  a.salariu,  r.cod\_hotel,  'receptionist',  l.cod\_locatie,  l.adresa  )  BULK COLLECT INTO v\_lista\_aux\_angajat  FROM receptionist r LEFT JOIN angajat a ON (r.cod\_angajat = a.cod\_angajat)  LEFT JOIN hotel h ON (r.cod\_hotel = h.cod\_hotel)  LEFT JOIN locatie l ON (h.cod\_locatie = l.cod\_locatie);    FOR i IN v\_lista\_aux\_angajat.FIRST..v\_lista\_aux\_angajat.LAST  LOOP  v\_deja\_luat := FALSE;    IF v\_lista\_angajat.COUNT > 0 THEN  FOR j IN v\_lista\_angajat.FIRST..v\_lista\_angajat.LAST  LOOP  IF v\_lista\_aux\_angajat(i).cod\_angajat = v\_lista\_angajat(j).cod\_angajat THEN  v\_deja\_luat := TRUE;  END IF;  END LOOP;  END IF;    IF v\_deja\_luat = FALSE THEN  v\_lista\_angajat.EXTEND;  v\_lista\_angajat(v\_lista\_angajat.COUNT) := v\_lista\_aux\_angajat(i);  END IF;  END LOOP;  --administrator  SELECT tip\_ex8(  ad.cod\_angajat,  a.nume,  a.prenume,  a.salariu,  ad.cod\_hotel,  'administrator',  l.cod\_locatie,  l.adresa  )  BULK COLLECT INTO v\_lista\_aux\_angajat  FROM administrator ad LEFT JOIN angajat a ON (ad.cod\_angajat = a.cod\_angajat)  LEFT JOIN hotel h ON (ad.cod\_hotel = h.cod\_hotel)  LEFT JOIN locatie l ON (h.cod\_locatie = l.cod\_locatie);    FOR i IN v\_lista\_aux\_angajat.FIRST..v\_lista\_aux\_angajat.LAST  LOOP  v\_deja\_luat := FALSE;    IF v\_lista\_angajat.COUNT > 0 THEN  FOR j IN v\_lista\_angajat.FIRST..v\_lista\_angajat.LAST  LOOP  IF v\_lista\_aux\_angajat(i).cod\_angajat = v\_lista\_angajat(j).cod\_angajat THEN  v\_deja\_luat := TRUE;  END IF;  END LOOP;  END IF;    IF v\_deja\_luat = FALSE THEN  v\_lista\_angajat.EXTEND;  v\_lista\_angajat(v\_lista\_angajat.COUNT) := v\_lista\_aux\_angajat(i);  END IF;  END LOOP;  --chelner  SELECT tip\_ex8(  c.cod\_angajat,  a.nume,  a.prenume,  a.salariu,  c.cod\_restaurant,  'chelner',  l.cod\_locatie,  l.adresa  )  BULK COLLECT INTO v\_lista\_aux\_angajat  FROM chelner c LEFT JOIN angajat a ON (c.cod\_angajat = a.cod\_angajat)  LEFT JOIN restaurant r ON (c.cod\_restaurant = r.cod\_restaurant)  LEFT JOIN locatie l ON (r.cod\_locatie = l.cod\_locatie);    FOR i IN v\_lista\_aux\_angajat.FIRST..v\_lista\_aux\_angajat.LAST  LOOP  v\_deja\_luat := FALSE;    IF v\_lista\_angajat.COUNT > 0 THEN  FOR j IN v\_lista\_angajat.FIRST..v\_lista\_angajat.LAST  LOOP  IF v\_lista\_aux\_angajat(i).cod\_angajat = v\_lista\_angajat(j).cod\_angajat THEN  v\_deja\_luat := TRUE;  END IF;  END LOOP;  END IF;    IF v\_deja\_luat = FALSE THEN  v\_lista\_angajat.EXTEND;  v\_lista\_angajat(v\_lista\_angajat.COUNT) := v\_lista\_aux\_angajat(i);  END IF;  END LOOP;  --bucatar  SELECT tip\_ex8(  b.cod\_angajat,  a.nume,  a.prenume,  a.salariu,  b.cod\_restaurant,  'bucatar',  l.cod\_locatie,  l.adresa  )  BULK COLLECT INTO v\_lista\_aux\_angajat  FROM bucatar b LEFT JOIN angajat a ON (b.cod\_angajat = a.cod\_angajat)  LEFT JOIN restaurant r ON (b.cod\_restaurant = r.cod\_restaurant)  LEFT JOIN locatie l ON (r.cod\_locatie = l.cod\_locatie);    FOR i IN v\_lista\_aux\_angajat.FIRST..v\_lista\_aux\_angajat.LAST  LOOP  v\_deja\_luat := FALSE;    IF v\_lista\_angajat.COUNT > 0 THEN  FOR j IN v\_lista\_angajat.FIRST..v\_lista\_angajat.LAST  LOOP  IF v\_lista\_aux\_angajat(i).cod\_angajat = v\_lista\_angajat(j).cod\_angajat THEN  v\_deja\_luat := TRUE;  END IF;  END LOOP;  END IF;    IF v\_deja\_luat = FALSE THEN  v\_lista\_angajat.EXTEND;  v\_lista\_angajat(v\_lista\_angajat.COUNT) := v\_lista\_aux\_angajat(i);  END IF;  END LOOP;    --mai trebuie selectati exact o data angajatii care nu lucreaza pe niciun post din cele mentionate  --(paznic, receptionist, administrator, chelner, bucatar)  FOR i IN (SELECT cod\_angajat, nume, prenume, salariu FROM angajat)  LOOP  v\_deja\_luat := FALSE;    FOR j IN v\_lista\_angajat.FIRST..v\_lista\_angajat.LAST  LOOP  IF v\_lista\_angajat(j).cod\_angajat = i.cod\_angajat THEN  v\_deja\_luat := TRUE;  END IF;  END LOOP;    IF v\_deja\_luat = FALSE THEN  v\_lista\_angajat.EXTEND;  v\_lista\_angajat(v\_lista\_angajat.COUNT) := tip\_ex8(i.cod\_angajat, i.nume, i.prenume, i.salariu, null, '', null, null);  END IF;    END LOOP;    v\_sortat := FALSE;    WHILE v\_sortat = FALSE --BubbleSort, sortez dupa salariu lista tuturor angajatilor  LOOP  v\_sortat := TRUE;    FOR i IN 1..(v\_lista\_angajat.COUNT - 1)  LOOP  IF v\_lista\_angajat(i).salariu < v\_lista\_angajat(i + 1).salariu THEN  v\_sortat := FALSE;  v\_aux\_angajat := v\_lista\_angajat(i);  v\_lista\_angajat(i) := v\_lista\_angajat(i + 1);  v\_lista\_angajat(i + 1) := v\_aux\_angajat;  END IF;  END LOOP;    END LOOP;    RETURN v\_lista\_angajat(N);  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista angajati in baza de date');  RETURN NULL;  WHEN N\_INVALID THEN  DBMS\_OUTPUT.PUT\_LINE('Valoarea N data ca parametru functiei este un numar invalid (negativ sau 0)');  RETURN NULL;  WHEN N\_PREA\_MARE THEN  DBMS\_OUTPUT.PUT\_LINE('Valoarea N data ca parametru functiei este un numar mai mare decat numarul total de angajati');  RETURN NULL;  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Alte erori');  RETURN NULL;  END ex8;  /    DECLARE  --v\_n NUMBER(10) := &p\_n;  v\_rezultat tip\_ex8;  BEGIN  v\_rezultat := ex8(-1);  DBMS\_OUTPUT.PUT\_LINE('Angajatul ' || v\_rezultat.cod\_angajat || ' ' || v\_rezultat.nume || ' '  || v\_rezultat.prenume || ', avand salariul ' || v\_rezultat.salariu || ', lucreaza la ' || v\_rezultat.cod\_munca  || ', pe post de ' || v\_rezultat.tip\_munca || ', locatia muncii fiind ' || v\_rezultat.cod\_locatie || ' ' ||  v\_rezultat.adresa);    v\_rezultat := ex8(0);  DBMS\_OUTPUT.PUT\_LINE('Angajatul ' || v\_rezultat.cod\_angajat || ' ' || v\_rezultat.nume || ' '  || v\_rezultat.prenume || ', avand salariul ' || v\_rezultat.salariu || ', lucreaza la ' || v\_rezultat.cod\_munca  || ', pe post de ' || v\_rezultat.tip\_munca || ', locatia muncii fiind ' || v\_rezultat.cod\_locatie || ' ' ||  v\_rezultat.adresa);    v\_rezultat := ex8(4);  DBMS\_OUTPUT.PUT\_LINE('Angajatul ' || v\_rezultat.cod\_angajat || ' ' || v\_rezultat.nume || ' '  || v\_rezultat.prenume || ', avand salariul ' || v\_rezultat.salariu || ', lucreaza la ' || v\_rezultat.cod\_munca  || ', pe post de ' || v\_rezultat.tip\_munca || ', locatia muncii fiind ' || v\_rezultat.cod\_locatie || ' ' ||  v\_rezultat.adresa);    v\_rezultat := ex8(101);  DBMS\_OUTPUT.PUT\_LINE('Angajatul ' || v\_rezultat.cod\_angajat || ' ' || v\_rezultat.nume || ' '  || v\_rezultat.prenume || ', avand salariul ' || v\_rezultat.salariu || ', lucreaza la ' || v\_rezultat.cod\_munca  || ', pe post de ' || v\_rezultat.tip\_munca || ', locatia muncii fiind ' || v\_rezultat.cod\_locatie || ' ' ||  v\_rezultat.adresa);  END;  /    ROLLBACK;  COMMIT;    DELETE FROM paznic;  DELETE FROM receptionist;  DELETE FROM administrator;  DELETE FROM chelner;  DELETE FROM bucatar;    DELETE FROM angajat; |









# Exercitiul 9

Formulati in limbaj natural o problema pe care sa o rezolvati folosind un subprogram stocat independent de tip procedura care sa utilizeze intr-o singura comanda SQL 5 dintre tabelele definite. Tratati toate exceptiile care pot aparea, incluzand exceptiile NO\_DATA\_FOUND si TOO\_MANY\_ROWS. Apelati subprogramul astfel incat sa evidentiati toate cazurile tratate.

Cerinta:

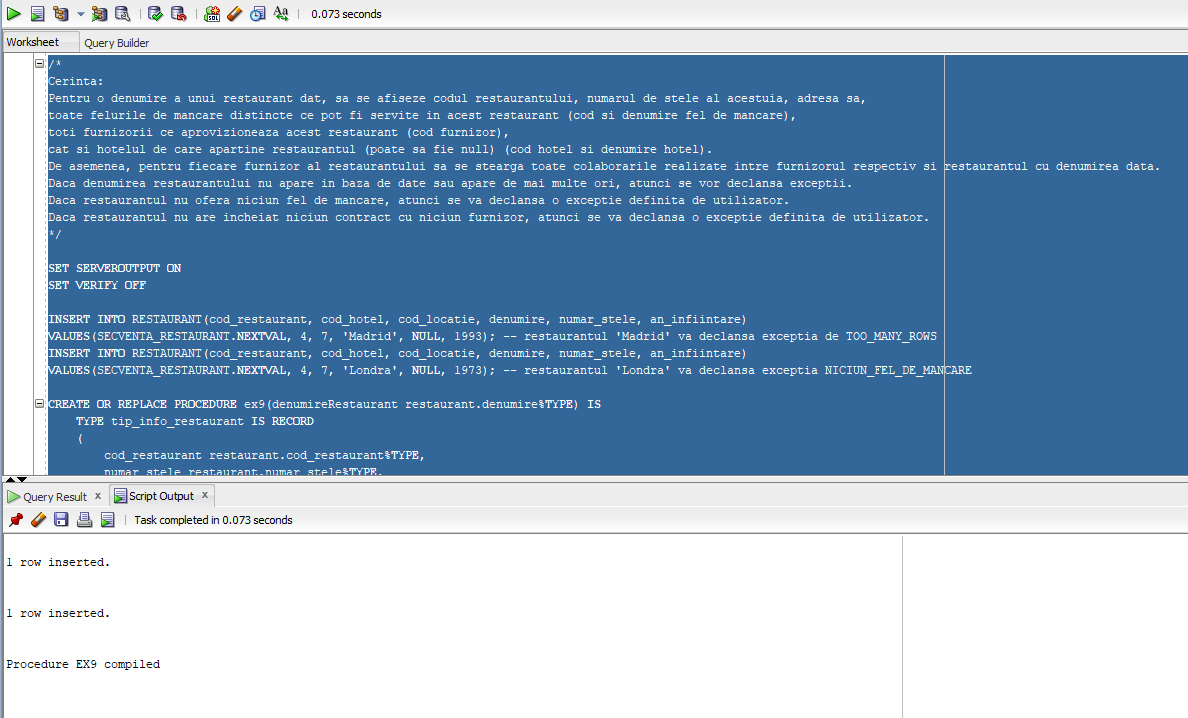
Pentru o denumire a unui restaurant dat, sa se afiseze codul restaurantului, numarul de stele al acestuia, adresa sa, toate felurile de mancare distincte ce pot fi servite in acest restaurant (cod si denumire fel de mancare), toti furnizorii ce aprovizioneaza acest restaurant (cod furnizor), cat si hotelul de care apartine restaurantul (poate sa fie null) (cod hotel si denumire hotel). De asemenea, pentru fiecare furnizor al restaurantului sa se stearga toate colaborarile realizate intre furnizorul respectiv si restaurantul cu denumirea data.

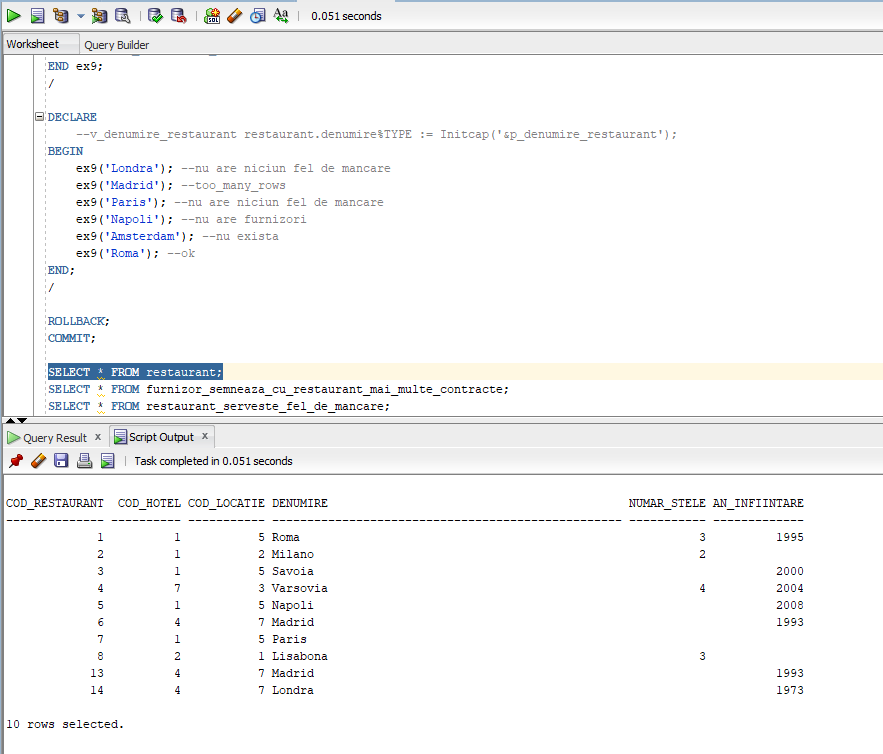
Daca denumirea restaurantului nu apare in baza de date sau apare de mai multe ori, atunci se vor declansa exceptii.

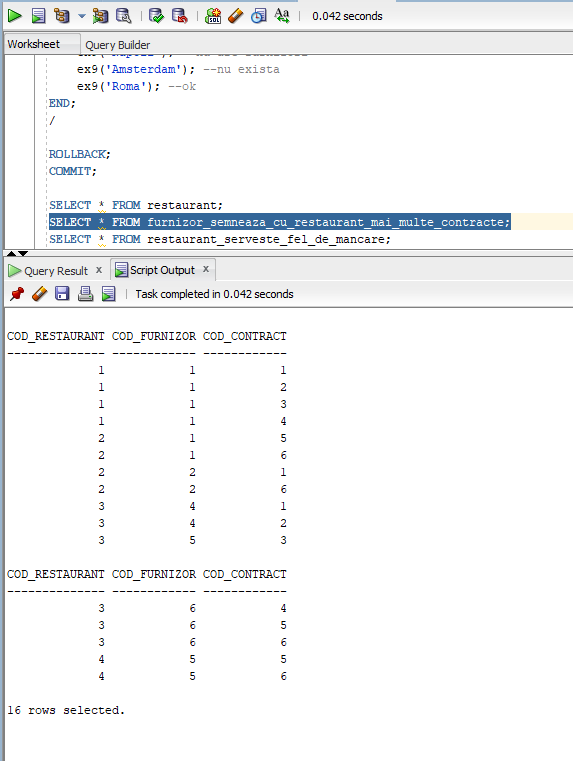
Daca restaurantul nu ofera niciun fel de mancare, atunci se va declansa o exceptie definita de utilizator.

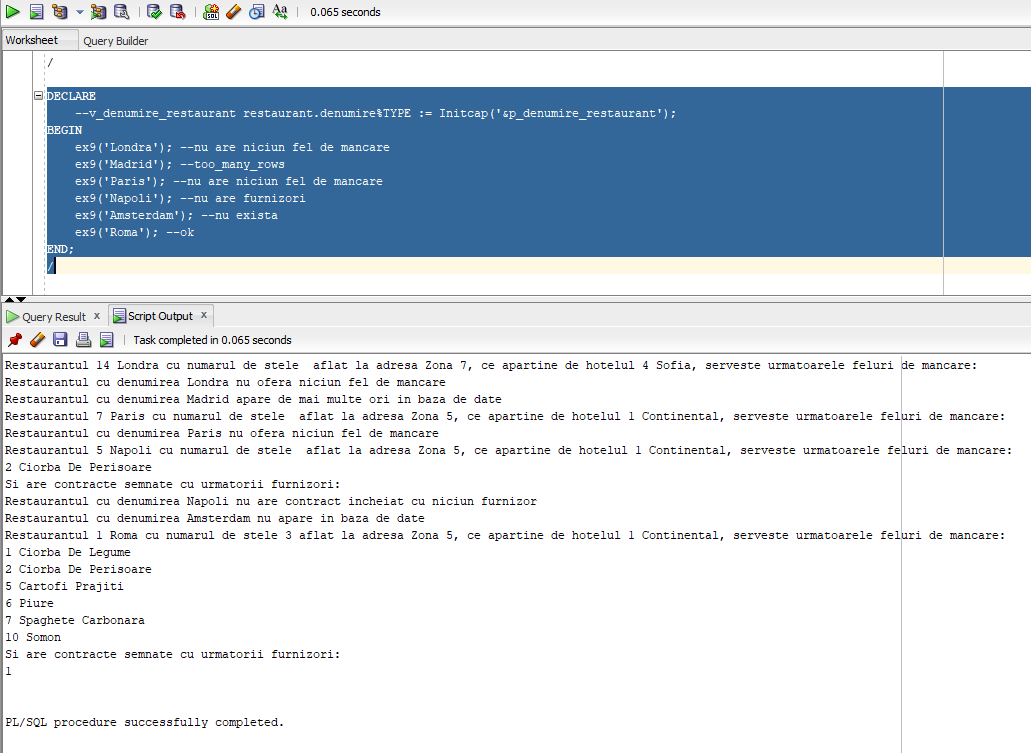
Daca restaurantul nu are incheiat niciun contract cu niciun furnizor, atunci se va declansa o exceptie definita de utilizator.

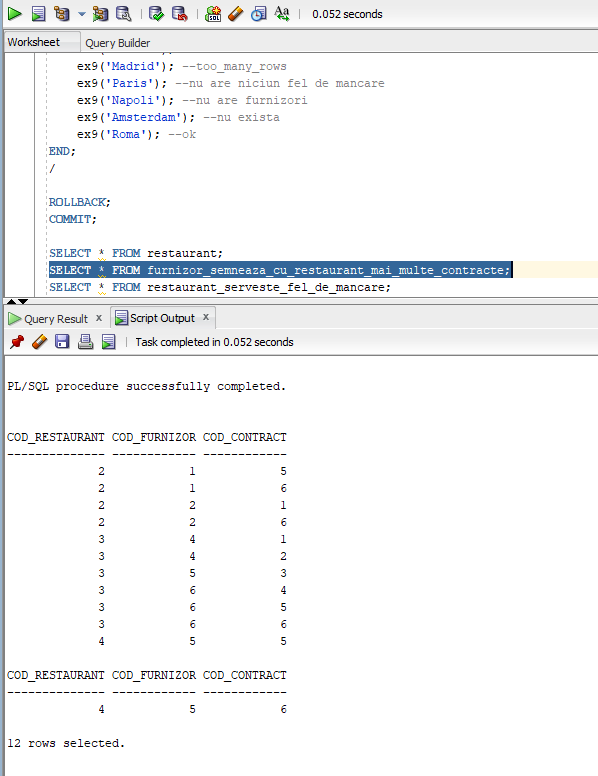
|  |
| --- |
| /\*  Cerinta:  Pentru o denumire a unui restaurant dat, sa se afiseze codul restaurantului, numarul de stele al acestuia, adresa sa,  toate felurile de mancare distincte ce pot fi servite in acest restaurant (cod si denumire fel de mancare),  toti furnizorii ce aprovizioneaza acest restaurant (cod furnizor),  cat si hotelul de care apartine restaurantul (poate sa fie null) (cod hotel si denumire hotel).  De asemenea, pentru fiecare furnizor al restaurantului sa se stearga toate colaborarile realizate intre furnizorul respectiv si restaurantul cu denumirea data.  Daca denumirea restaurantului nu apare in baza de date sau apare de mai multe ori, atunci se vor declansa exceptii.  Daca restaurantul nu ofera niciun fel de mancare, atunci se va declansa o exceptie definita de utilizator.  Daca restaurantul nu are incheiat niciun contract cu niciun furnizor, atunci se va declansa o exceptie definita de utilizator.  \*/    SET SERVEROUTPUT ON  SET VERIFY OFF    INSERT INTO RESTAURANT(cod\_restaurant, cod\_hotel, cod\_locatie, denumire, numar\_stele, an\_infiintare)  VALUES(SECVENTA\_RESTAURANT.NEXTVAL, 4, 7, 'Madrid', NULL, 1993); -- restaurantul 'Madrid' va declansa exceptia de TOO\_MANY\_ROWS  INSERT INTO RESTAURANT(cod\_restaurant, cod\_hotel, cod\_locatie, denumire, numar\_stele, an\_infiintare)  VALUES(SECVENTA\_RESTAURANT.NEXTVAL, 4, 7, 'Londra', NULL, 1973); -- restaurantul 'Londra' va declansa exceptia NICIUN\_FEL\_DE\_MANCARE    CREATE OR REPLACE PROCEDURE ex9(denumireRestaurant restaurant.denumire%TYPE) IS  TYPE tip\_info\_restaurant IS RECORD  (  cod\_restaurant restaurant.cod\_restaurant%TYPE,  numar\_stele restaurant.numar\_stele%TYPE,  adresa locatie.adresa%TYPE,  cod\_fel\_de\_mancare fel\_de\_mancare.cod\_fel\_de\_mancare%TYPE,  denumire\_fel\_de\_mancare fel\_de\_mancare.denumire%TYPE,  cod\_furnizor furnizor.cod\_furnizor%TYPE,  cod\_hotel hotel.cod\_hotel%TYPE,  denumire\_hotel hotel.denumire%TYPE  );  TYPE tip\_lista\_info\_restaurant IS VARRAY(1000) OF tip\_info\_restaurant;  v\_info\_restaurant tip\_info\_restaurant;  v\_lista\_info\_restaurant tip\_lista\_info\_restaurant := tip\_lista\_info\_restaurant();    v\_cod\_restaurant restaurant.cod\_restaurant%TYPE;    TYPE tip\_info\_fel\_de\_mancare IS RECORD  (  cod\_fel\_de\_mancare fel\_de\_mancare.cod\_fel\_de\_mancare%TYPE,  denumire fel\_de\_mancare.denumire%TYPE  );    TYPE tip\_tabel\_feluri\_de\_mancare IS TABLE OF tip\_info\_fel\_de\_mancare;  TYPE tip\_tabel\_furnizori IS TABLE OF furnizor.cod\_furnizor%TYPE;    tabel\_feluri\_de\_mancare tip\_tabel\_feluri\_de\_mancare := tip\_tabel\_feluri\_de\_mancare();  tabel\_furnizori tip\_tabel\_furnizori := tip\_tabel\_furnizori();    v\_deja\_exista BOOLEAN := FALSE;    CURSOR c\_info\_restaurant(codRestaurant restaurant.cod\_restaurant%TYPE)  IS  SELECT r.cod\_restaurant, r.numar\_stele, l.adresa, m.cod\_fel\_de\_mancare, m.denumire, f.cod\_furnizor, h.cod\_hotel, h.denumire  FROM restaurant r LEFT JOIN locatie l ON (r.cod\_locatie = l.cod\_locatie)  LEFT JOIN restaurant\_serveste\_fel\_de\_mancare rm ON (r.cod\_restaurant = rm.cod\_restaurant)  LEFT JOIN fel\_de\_mancare m ON (rm.cod\_fel\_de\_mancare = m.cod\_fel\_de\_mancare)  LEFT JOIN hotel h ON (r.cod\_hotel = h.cod\_hotel)  LEFT JOIN furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte frc ON (r.cod\_restaurant = frc.cod\_restaurant)  LEFT JOIN furnizor f ON (frc.cod\_furnizor = f.cod\_furnizor)  WHERE r.cod\_restaurant = codRestaurant;    NICIUN\_FEL\_DE\_MANCARE EXCEPTION;  NICIUN\_FURNIZOR EXCEPTION;    PRAGMA EXCEPTION\_INIT(NICIUN\_FEL\_DE\_MANCARE, -20001);  PRAGMA EXCEPTION\_INIT(NICIUN\_FURNIZOR, -20002);    BEGIN  SELECT cod\_restaurant INTO v\_cod\_restaurant --acest query va declansa exceptiile NO\_DATA\_FOUND sau TOO\_MANY\_ROWS  FROM restaurant WHERE INITCAP(denumire) = INITCAP(denumireRestaurant);    OPEN c\_info\_restaurant(v\_cod\_restaurant);    FETCH c\_info\_restaurant INTO v\_info\_restaurant;    WHILE c\_info\_restaurant%FOUND  LOOP  v\_lista\_info\_restaurant.EXTEND;  v\_lista\_info\_restaurant(v\_lista\_info\_restaurant.COUNT) := v\_info\_restaurant;  FETCH c\_info\_restaurant INTO v\_info\_restaurant;  END LOOP;    CLOSE c\_info\_restaurant;      FOR i IN v\_lista\_info\_restaurant.FIRST..v\_lista\_info\_restaurant.LAST  LOOP  IF v\_lista\_info\_restaurant(i).cod\_fel\_de\_mancare IS NOT NULL THEN    v\_deja\_exista := FALSE;    IF tabel\_feluri\_de\_mancare.COUNT > 0 THEN  FOR j IN tabel\_feluri\_de\_mancare.FIRST..tabel\_feluri\_de\_mancare.LAST  LOOP  IF tabel\_feluri\_de\_mancare(j).cod\_fel\_de\_mancare = v\_lista\_info\_restaurant(i).cod\_fel\_de\_mancare THEN  v\_deja\_exista := TRUE;  END IF;  END LOOP;  END IF;    IF v\_deja\_exista = FALSE THEN  tabel\_feluri\_de\_mancare.EXTEND;  tabel\_feluri\_de\_mancare(tabel\_feluri\_de\_mancare.LAST) := tip\_info\_fel\_de\_mancare(v\_lista\_info\_restaurant(i).cod\_fel\_de\_mancare, v\_lista\_info\_restaurant(i).denumire\_fel\_de\_mancare);  END IF;  END IF;    IF v\_lista\_info\_restaurant(i).cod\_furnizor IS NOT NULL THEN  IF v\_lista\_info\_restaurant(i).cod\_furnizor NOT MEMBER OF tabel\_furnizori THEN  tabel\_furnizori.EXTEND;  tabel\_furnizori(tabel\_furnizori.LAST) := v\_lista\_info\_restaurant(i).cod\_furnizor;  END IF;  END IF;  END LOOP;    DBMS\_OUTPUT.PUT\_LINE('Restaurantul ' || v\_lista\_info\_restaurant(1).cod\_restaurant || ' ' || denumireRestaurant  || ' cu numarul de stele ' || v\_lista\_info\_restaurant(1).numar\_stele  || ' aflat la adresa ' || v\_lista\_info\_restaurant(1).adresa || ', ce apartine de hotelul '  || v\_lista\_info\_restaurant(1).cod\_hotel || ' ' || v\_lista\_info\_restaurant(1).denumire\_hotel  || ', serveste urmatoarele feluri de mancare: ');      IF tabel\_feluri\_de\_mancare.COUNT > 0 THEN  FOR i IN tabel\_feluri\_de\_mancare.FIRST..tabel\_feluri\_de\_mancare.LAST  LOOP  DBMS\_OUTPUT.PUT\_LINE(tabel\_feluri\_de\_mancare(i).cod\_fel\_de\_mancare || ' ' || tabel\_feluri\_de\_mancare(i).denumire);  END LOOP;  ELSE  RAISE NICIUN\_FEL\_DE\_MANCARE;  END IF;    DBMS\_OUTPUT.PUT\_LINE('Si are contracte semnate cu urmatorii furnizori: ');    IF tabel\_furnizori.COUNT > 0 THEN  FOR i IN tabel\_furnizori.FIRST..tabel\_furnizori.LAST  LOOP  DBMS\_OUTPUT.PUT\_LINE(tabel\_furnizori(i));  END LOOP;  ELSE  RAISE NICIUN\_FURNIZOR;  END IF;    FORALL i IN tabel\_furnizori.FIRST..tabel\_furnizori.LAST  DELETE FROM furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte frc  WHERE frc.cod\_furnizor = tabel\_furnizori(i) AND frc.cod\_restaurant = v\_cod\_restaurant;    EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Restaurantul cu denumirea ' || denumireRestaurant || ' nu apare in baza de date');  WHEN TOO\_MANY\_ROWS THEN  DBMS\_OUTPUT.PUT\_LINE('Restaurantul cu denumirea ' || denumireRestaurant || ' apare de mai multe ori in baza de date');  WHEN NICIUN\_FEL\_DE\_MANCARE THEN  DBMS\_OUTPUT.PUT\_LINE('Restaurantul cu denumirea ' || denumireRestaurant || ' nu ofera niciun fel de mancare');  WHEN NICIUN\_FURNIZOR THEN  DBMS\_OUTPUT.PUT\_LINE('Restaurantul cu denumirea ' || denumireRestaurant || ' nu are contract incheiat cu niciun furnizor');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Alte erori');  END ex9;  /    DECLARE  --v\_denumire\_restaurant restaurant.denumire%TYPE := Initcap('&p\_denumire\_restaurant');  BEGIN  ex9('Londra'); --nu are niciun fel de mancare  ex9('Madrid'); --too\_many\_rows  ex9('Paris'); --nu are niciun fel de mancare  ex9('Napoli'); --nu are furnizori  ex9('Amsterdam'); --nu exista  ex9('Roma'); --ok  END;  /    ROLLBACK;  COMMIT;    SELECT \* FROM restaurant;  SELECT \* FROM furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte;  SELECT \* FROM restaurant\_serveste\_fel\_de\_mancare;  ALTER TRIGGER ex10 DISABLE;  ALTER TRIGGER ex10 ENABLE; |











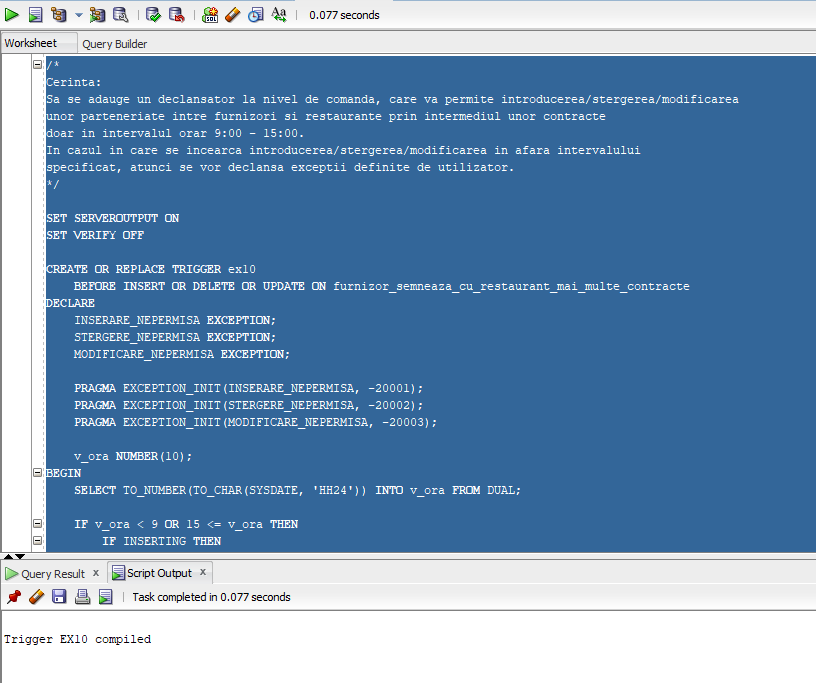
# Exercitiul 10

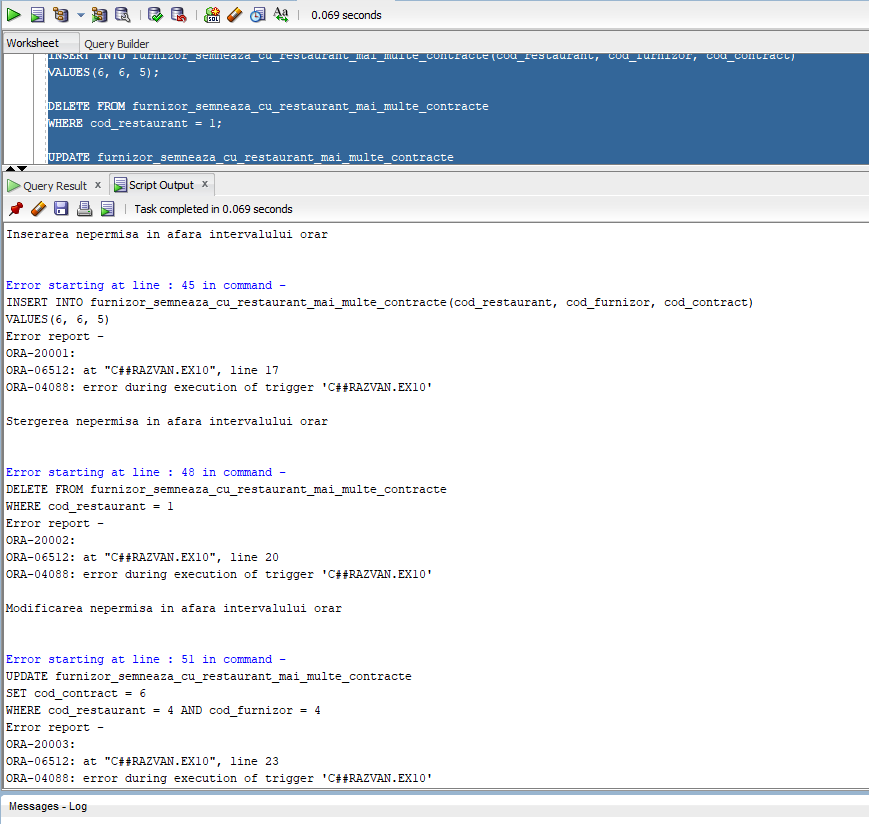
Definiti un trigger de tip LMD la nivel de comanda. Declansati trigger-ul.

Cerinta:

Sa se adauge un declansator la nivel de comanda, care va permite introducerea/stergerea/modificarea unor parteneriate intre furnizori si restaurante prin intermediul unor contracte doar in intervalul orar 9:00 - 15:00. In cazul in care se incearca introducerea/stergerea/modificarea in afara intervalului specificat, atunci se vor declansa exceptii definite de utilizator.

|  |
| --- |
| /\*  Cerinta:  Sa se adauge un declansator la nivel de comanda, care va permite introducerea/stergerea/modificarea  unor parteneriate intre furnizori si restaurante prin intermediul unor contracte  doar in intervalul orar 9:00 - 15:00.  In cazul in care se incearca introducerea/stergerea/modificarea in afara intervalului  specificat, atunci se vor declansa exceptii definite de utilizator.  \*/    SET SERVEROUTPUT ON  SET VERIFY OFF    CREATE OR REPLACE TRIGGER ex10  BEFORE INSERT OR DELETE OR UPDATE ON furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte  DECLARE  INSERARE\_NEPERMISA EXCEPTION;  STERGERE\_NEPERMISA EXCEPTION;  MODIFICARE\_NEPERMISA EXCEPTION;    PRAGMA EXCEPTION\_INIT(INSERARE\_NEPERMISA, -20001);  PRAGMA EXCEPTION\_INIT(STERGERE\_NEPERMISA, -20002);  PRAGMA EXCEPTION\_INIT(MODIFICARE\_NEPERMISA, -20003);    v\_ora NUMBER(10);  BEGIN  SELECT TO\_NUMBER(TO\_CHAR(SYSDATE, 'HH24')) INTO v\_ora FROM DUAL;    IF v\_ora < 9 OR 15 <= v\_ora THEN  IF INSERTING THEN  DBMS\_OUTPUT.PUT\_LINE('Inserarea nepermisa in afara intervalului orar');  RAISE INSERARE\_NEPERMISA;  ELSIF DELETING THEN  DBMS\_OUTPUT.PUT\_LINE('Stergerea nepermisa in afara intervalului orar');  RAISE STERGERE\_NEPERMISA;  ELSE -- UPDATING  DBMS\_OUTPUT.PUT\_LINE('Modificarea nepermisa in afara intervalului orar');  RAISE MODIFICARE\_NEPERMISA;  END IF;  END IF;  END ex10;  /    DESC furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte;    INSERT INTO furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte(cod\_restaurant, cod\_furnizor, cod\_contract)  VALUES(6, 6, 5);    DELETE FROM furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte  WHERE cod\_restaurant = 1;    UPDATE furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte  SET cod\_contract = 6  WHERE cod\_restaurant = 4 AND cod\_furnizor = 4;    SELECT \* FROM furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte;    ROLLBACK;  COMMIT; |





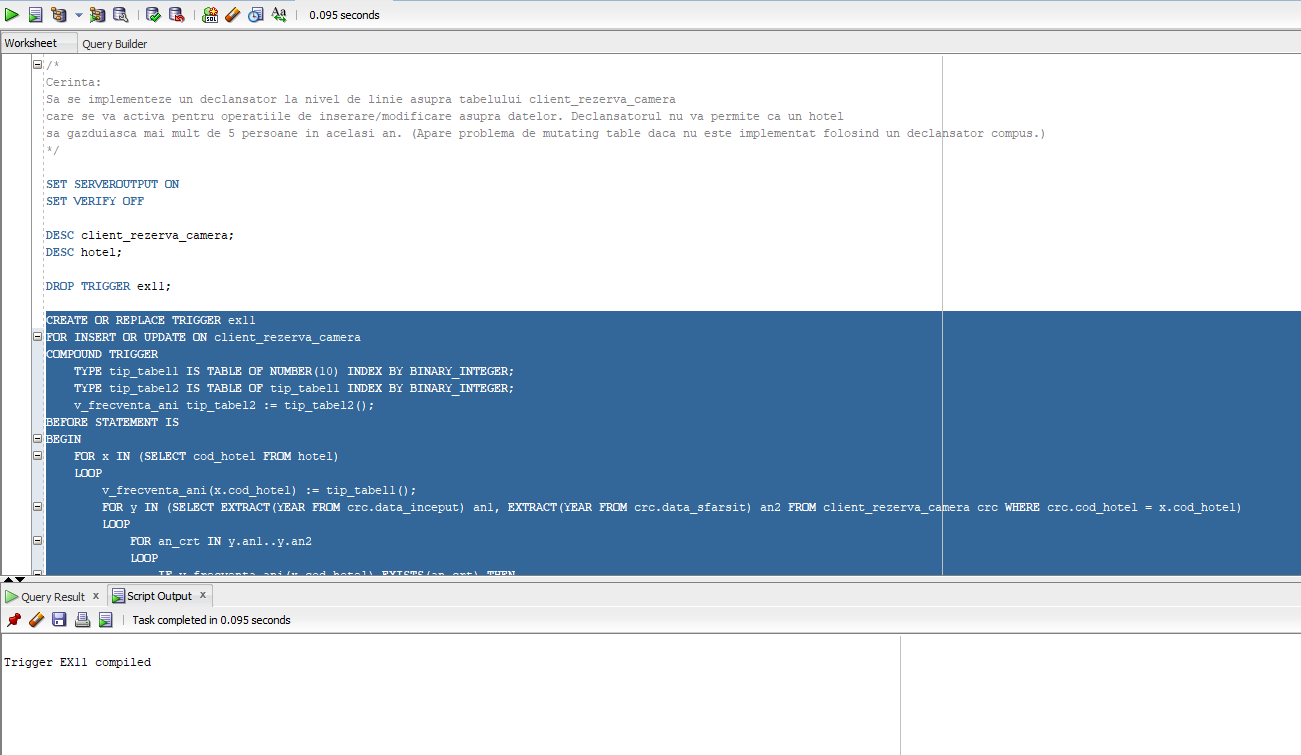
# Exercitiul 11

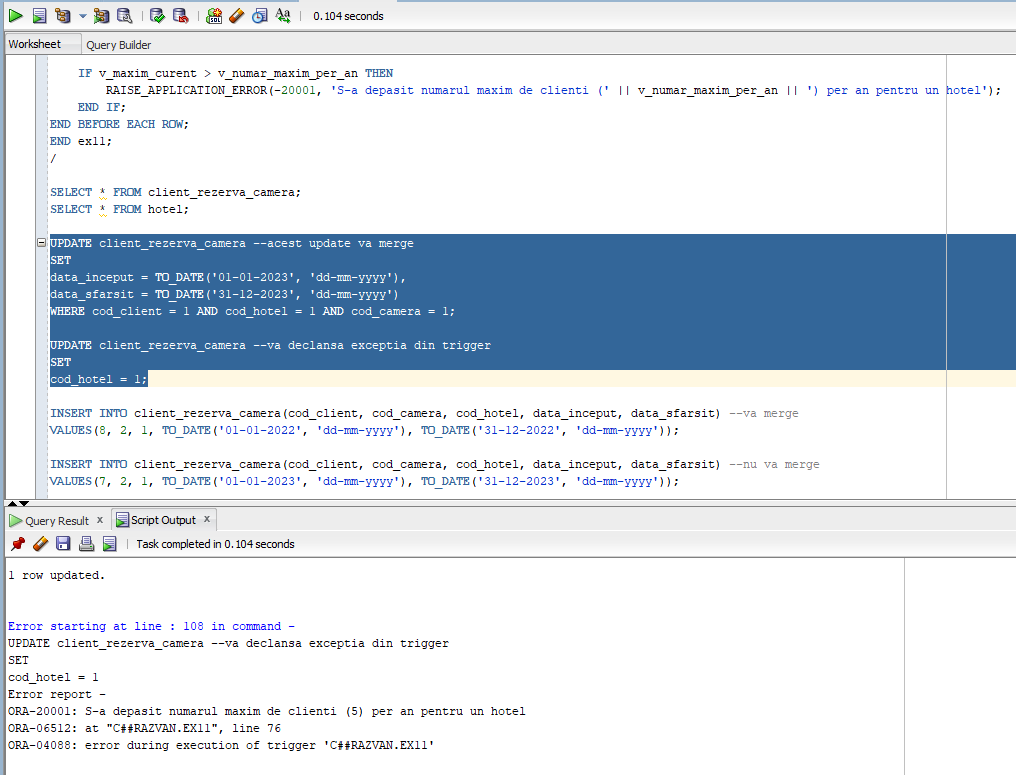
Definiti un trigger de tip LMD la nivel de linie. Declansati trigger-ul.

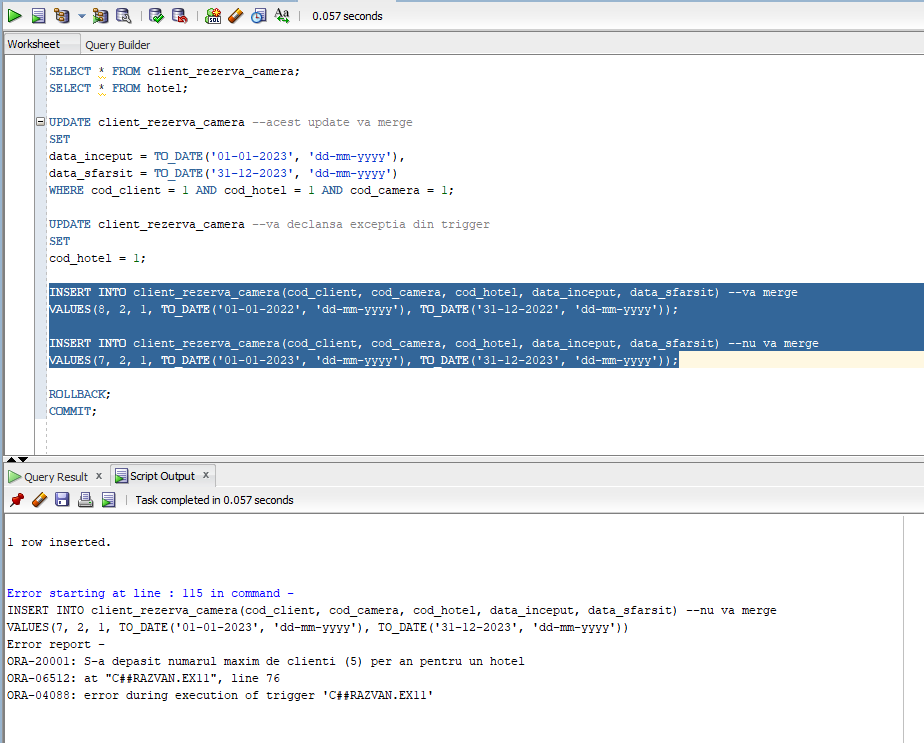
Cerinta:

Sa se implementeze un declansator la nivel de linie asupra tabelului client\_rezerva\_camera care se va activa pentru operatiile de inserare/modificare asupra datelor. Declansatorul nu va permite ca un hotel sa gazduiasca mai mult de 5 persoane in acelasi an. (Apare problema de mutating table daca nu este implementat folosind un declansator compus.)

|  |
| --- |
| /\*  Cerinta:  Sa se implementeze un declansator la nivel de linie asupra tabelului client\_rezerva\_camera  care se va activa pentru operatiile de inserare/modificare asupra datelor. Declansatorul nu va permite ca un hotel  sa gazduiasca mai mult de 5 persoane in acelasi an. (Apare problema de mutating table daca nu este implementat folosind un declansator compus.)  \*/    SET SERVEROUTPUT ON  SET VERIFY OFF    DESC client\_rezerva\_camera;  DESC hotel;    DROP TRIGGER ex11;    CREATE OR REPLACE TRIGGER ex11  FOR INSERT OR UPDATE ON client\_rezerva\_camera  COMPOUND TRIGGER  TYPE tip\_tabel1 IS TABLE OF NUMBER(10) INDEX BY BINARY\_INTEGER;  TYPE tip\_tabel2 IS TABLE OF tip\_tabel1 INDEX BY BINARY\_INTEGER;  v\_frecventa\_ani tip\_tabel2 := tip\_tabel2();  BEFORE STATEMENT IS  BEGIN  FOR x IN (SELECT cod\_hotel FROM hotel)  LOOP  v\_frecventa\_ani(x.cod\_hotel) := tip\_tabel1();  FOR y IN (SELECT EXTRACT(YEAR FROM crc.data\_inceput) an1, EXTRACT(YEAR FROM crc.data\_sfarsit) an2 FROM client\_rezerva\_camera crc WHERE crc.cod\_hotel = x.cod\_hotel)  LOOP  FOR an\_crt IN y.an1..y.an2  LOOP  IF v\_frecventa\_ani(x.cod\_hotel).EXISTS(an\_crt) THEN  v\_frecventa\_ani(x.cod\_hotel)(an\_crt) := v\_frecventa\_ani(x.cod\_hotel)(an\_crt) + 1;  ELSE  v\_frecventa\_ani(x.cod\_hotel)(an\_crt) := 1;  END IF;  END LOOP;  END LOOP;  END LOOP;  END BEFORE STATEMENT;  BEFORE EACH ROW IS  v\_an1 NUMBER(10) := 0;  v\_an2 NUMBER(10) := 0;  v\_numar\_maxim\_per\_an NUMBER(10) := 5;  v\_maxim\_curent NUMBER(10) := 0;  BEGIN  IF INSERTING THEN  v\_an1 := EXTRACT(YEAR FROM :NEW.data\_inceput);  v\_an2 := EXTRACT(YEAR FROM :NEW.data\_sfarsit);    FOR an\_crt IN v\_an1..v\_an2  LOOP  IF v\_frecventa\_ani(:NEW.cod\_hotel).EXISTS(an\_crt) THEN  v\_frecventa\_ani(:NEW.cod\_hotel)(an\_crt) := v\_frecventa\_ani(:NEW.cod\_hotel)(an\_crt) + 1;  ELSE  v\_frecventa\_ani(:NEW.cod\_hotel)(an\_crt) := 1;  END IF;  END LOOP;  ELSE -- UPDATING  v\_an1 := EXTRACT(YEAR FROM :OLD.data\_inceput);  v\_an2 := EXTRACT(YEAR FROM :OLD.data\_sfarsit);    FOR an\_crt IN v\_an1..v\_an2  LOOP  v\_frecventa\_ani(:OLD.cod\_hotel)(an\_crt) := v\_frecventa\_ani(:OLD.cod\_hotel)(an\_crt) - 1;  END LOOP;    v\_an1 := EXTRACT(YEAR FROM :NEW.data\_inceput);  v\_an2 := EXTRACT(YEAR FROM :NEW.data\_sfarsit);    FOR an\_crt IN v\_an1..v\_an2  LOOP  IF v\_frecventa\_ani(:NEW.cod\_hotel).EXISTS(an\_crt) THEN  v\_frecventa\_ani(:NEW.cod\_hotel)(an\_crt) := v\_frecventa\_ani(:NEW.cod\_hotel)(an\_crt) + 1;  ELSE  v\_frecventa\_ani(:NEW.cod\_hotel)(an\_crt) := 1;  END IF;  END LOOP;  END IF;    FOR x IN (SELECT cod\_hotel FROM hotel)  LOOP  IF v\_frecventa\_ani(x.cod\_hotel).COUNT > 0 THEN  FOR y IN v\_frecventa\_ani(x.cod\_hotel).FIRST..v\_frecventa\_ani(x.cod\_hotel).LAST  LOOP  IF v\_maxim\_curent < v\_frecventa\_ani(x.cod\_hotel)(y) THEN  v\_maxim\_curent := v\_frecventa\_ani(x.cod\_hotel)(y);  END IF;  END LOOP;  END IF;  END LOOP;    IF v\_maxim\_curent > v\_numar\_maxim\_per\_an THEN  RAISE\_APPLICATION\_ERROR(-20001, 'S-a depasit numarul maxim de clienti (' || v\_numar\_maxim\_per\_an || ') per an pentru un hotel');  END IF;  END BEFORE EACH ROW;  END ex11;  /    SELECT \* FROM client\_rezerva\_camera;  SELECT \* FROM hotel;    UPDATE client\_rezerva\_camera --acest update va merge  SET  data\_inceput = TO\_DATE('01-01-2023', 'dd-mm-yyyy'),  data\_sfarsit = TO\_DATE('31-12-2023', 'dd-mm-yyyy')  WHERE cod\_client = 1 AND cod\_hotel = 1 AND cod\_camera = 1;    UPDATE client\_rezerva\_camera --va declansa exceptia din trigger  SET  cod\_hotel = 1;    INSERT INTO client\_rezerva\_camera(cod\_client, cod\_camera, cod\_hotel, data\_inceput, data\_sfarsit) --va merge  VALUES(8, 2, 1, TO\_DATE('01-01-2022', 'dd-mm-yyyy'), TO\_DATE('31-12-2022', 'dd-mm-yyyy'));    INSERT INTO client\_rezerva\_camera(cod\_client, cod\_camera, cod\_hotel, data\_inceput, data\_sfarsit) --nu va merge  VALUES(7, 2, 1, TO\_DATE('01-01-2023', 'dd-mm-yyyy'), TO\_DATE('31-12-2023', 'dd-mm-yyyy'));    ROLLBACK;  COMMIT; |







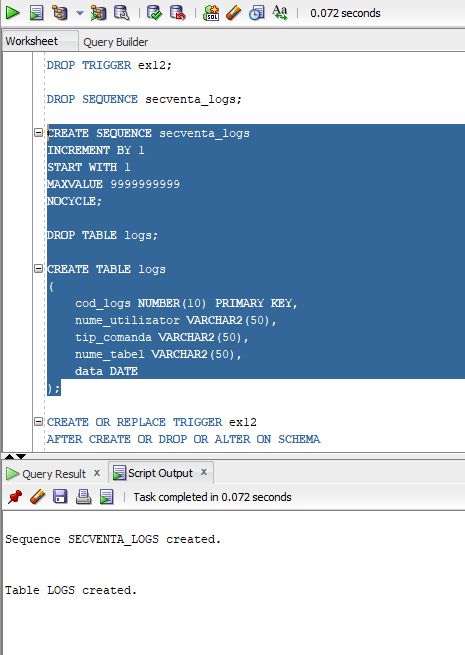
# Exercitiul 12

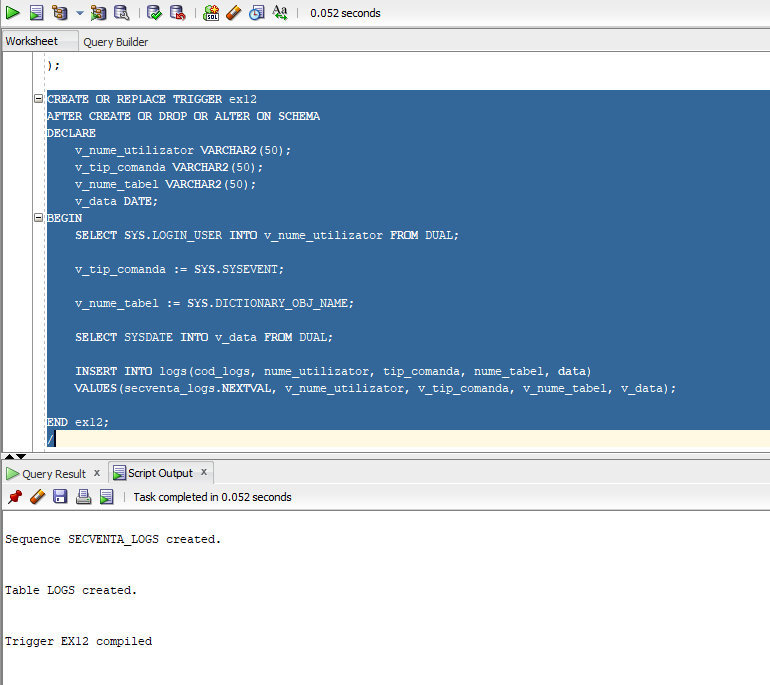
Definiti un trigger de tip LDD. Declansati trigger-ul.

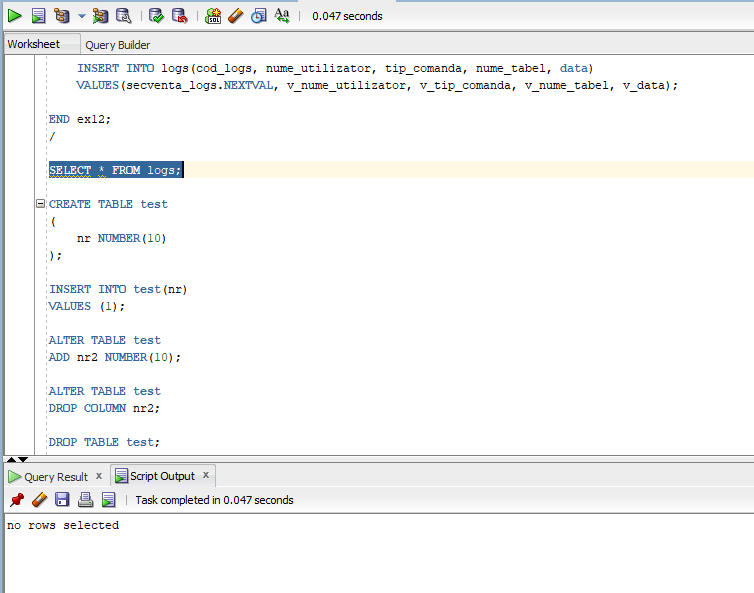
Cerinta:

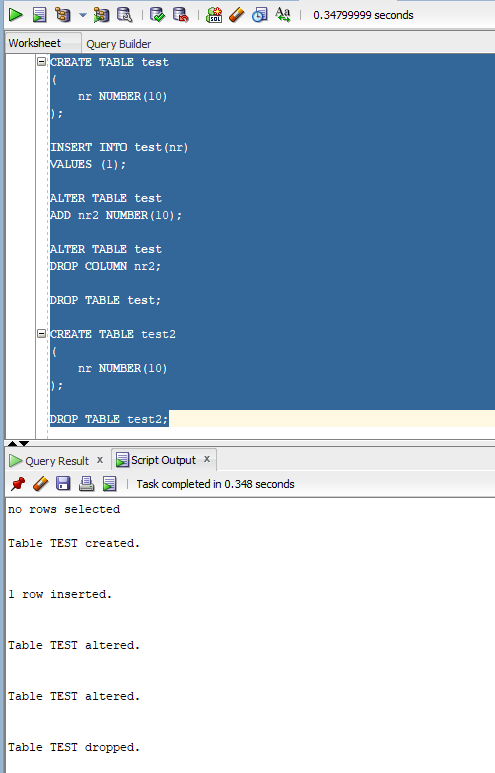
Sa se creeze un tabel de logs, care va monitoriza modificarile realizate de utilizatori asupra bazei de date. (Operatii de create table, drop table, alter table) Tabelul de logs va memora numele utilizatorului care a efectuat comanda, tipul comenzii, tabelul asupra careia a fost aplicata si data cand s-a aplicat. Se va implementa un trigger asupra bazei de date care va insera in tabelul de logs de fiecare data cand o asemenea comanda va fi executata.

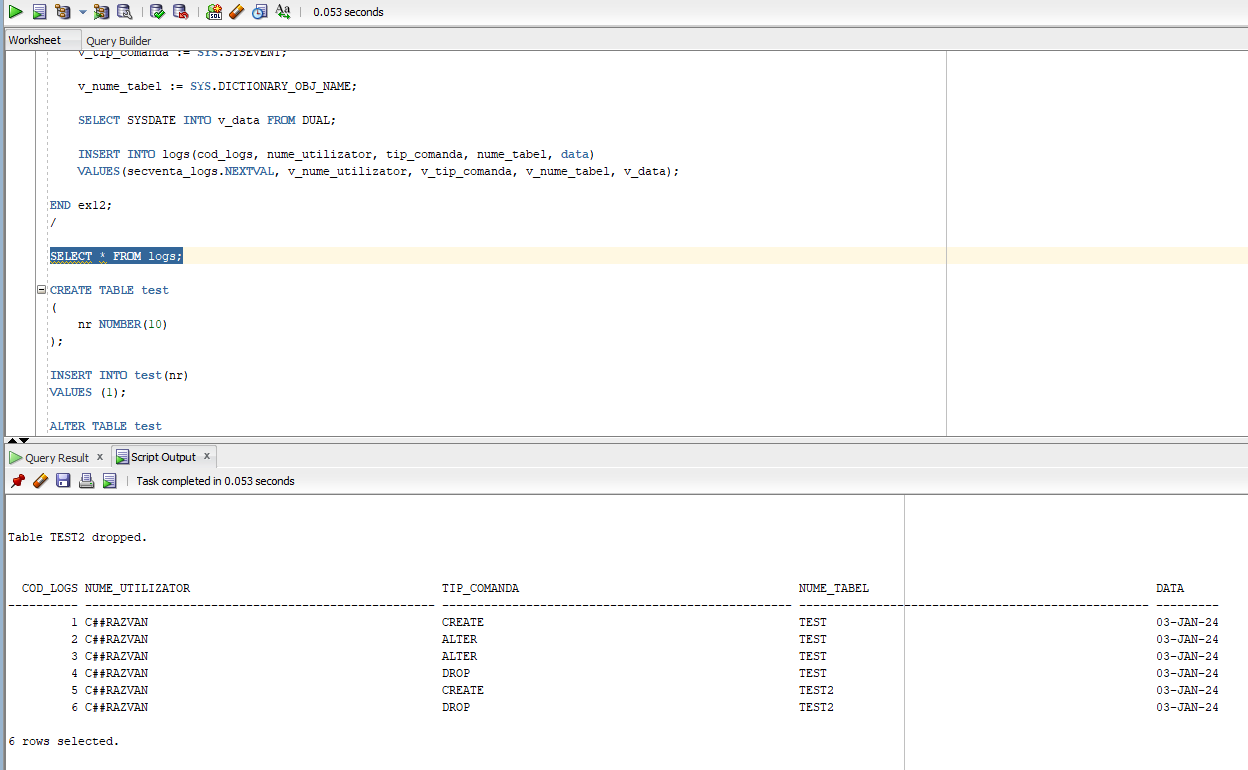
|  |
| --- |
| /\*  Cerinta:  Sa se creeze un tabel de logs, care va monitoriza modificarile realizate de utilizatori asupra bazei de date. (Operatii de create table, drop table, alter table)  Tabelul de logs va memora numele utilizatorului care a efectuat comanda, tipul comenzii, tabelul asupra careia a fost aplicata si data cand s-a aplicat.  Se va implementa un trigger asupra bazei de date care va insera in tabelul de logs de fiecare data cand o asemenea comanda va fi executata.  \*/    SET SERVEROUTPUT ON  SET VERIFY OFF    DROP TRIGGER ex12;    DROP SEQUENCE secventa\_logs;    CREATE SEQUENCE secventa\_logs  INCREMENT BY 1  START WITH 1  MAXVALUE 9999999999  NOCYCLE;    DROP TABLE logs;    CREATE TABLE logs  (  cod\_logs NUMBER(10) PRIMARY KEY,  nume\_utilizator VARCHAR2(50),  tip\_comanda VARCHAR2(50),  nume\_tabel VARCHAR2(50),  data DATE  );    CREATE OR REPLACE TRIGGER ex12  AFTER CREATE OR DROP OR ALTER ON SCHEMA  DECLARE  v\_nume\_utilizator VARCHAR2(50);  v\_tip\_comanda VARCHAR2(50);  v\_nume\_tabel VARCHAR2(50);  v\_data DATE;  BEGIN  SELECT SYS.LOGIN\_USER INTO v\_nume\_utilizator FROM DUAL;    v\_tip\_comanda := SYS.SYSEVENT;    v\_nume\_tabel := SYS.DICTIONARY\_OBJ\_NAME;    SELECT SYSDATE INTO v\_data FROM DUAL;    INSERT INTO logs(cod\_logs, nume\_utilizator, tip\_comanda, nume\_tabel, data)  VALUES(secventa\_logs.NEXTVAL, v\_nume\_utilizator, v\_tip\_comanda, v\_nume\_tabel, v\_data);    END ex12;  /    SELECT \* FROM logs;    CREATE TABLE test  (  nr NUMBER(10)  );    INSERT INTO test(nr)  VALUES (1);    ALTER TABLE test  ADD nr2 NUMBER(10);    ALTER TABLE test  DROP COLUMN nr2;    DROP TABLE test;    CREATE TABLE test2  (  nr NUMBER(10)  );    DROP TABLE test2;    ROLLBACK;  COMMIT; |







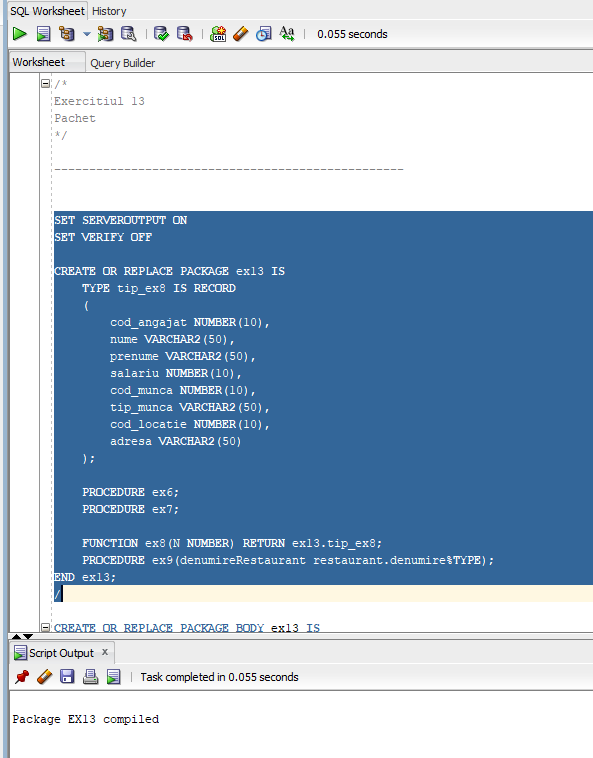




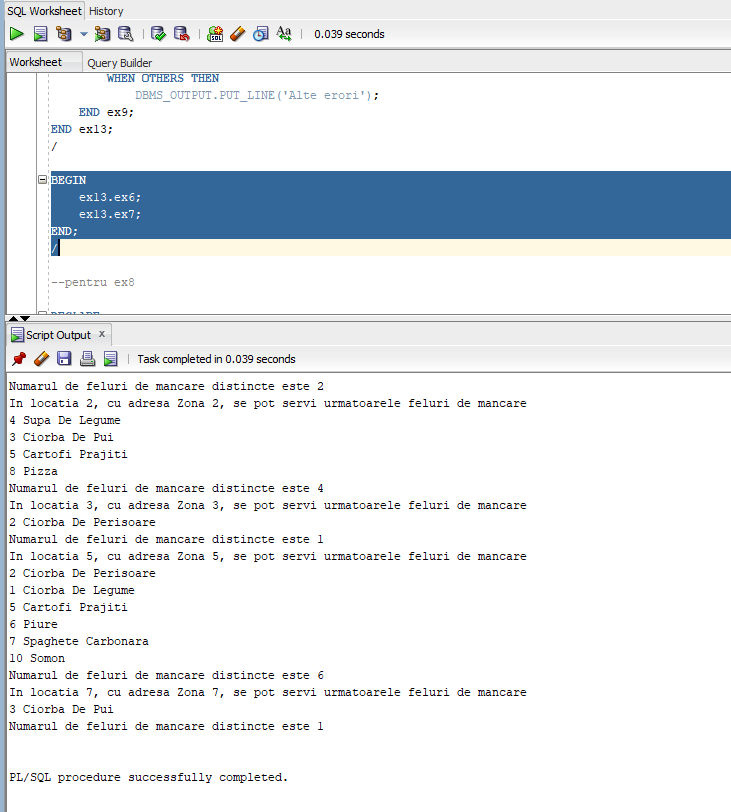
# Exercitiul 13

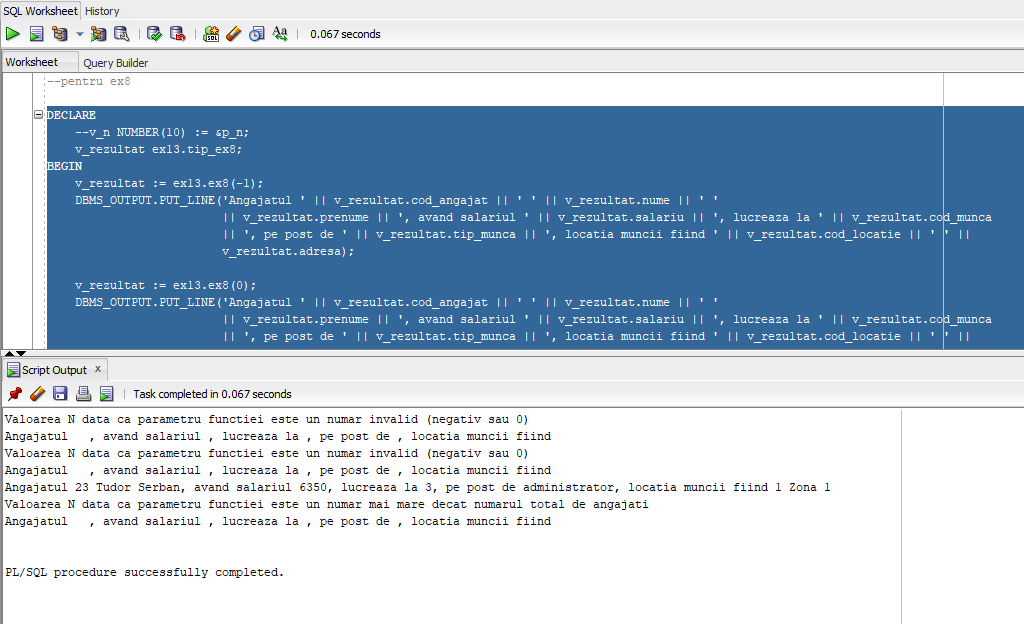
Definiti un pachet care sa contina toate obiectele definite in cadrul proiectului.

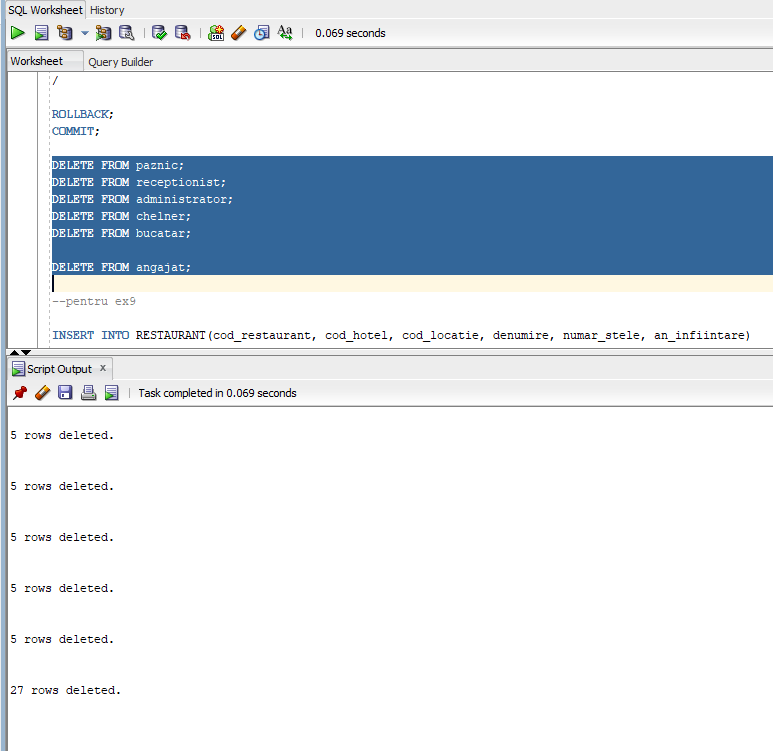
|  |
| --- |
| SET SERVEROUTPUT ON  SET VERIFY OFF    CREATE OR REPLACE PACKAGE ex13 IS  TYPE tip\_ex8 IS RECORD  (  cod\_angajat NUMBER(10),  nume VARCHAR2(50),  prenume VARCHAR2(50),  salariu NUMBER(10),  cod\_munca NUMBER(10),  tip\_munca VARCHAR2(50),  cod\_locatie NUMBER(10),  adresa VARCHAR2(50)  );    PROCEDURE ex6;  PROCEDURE ex7;    FUNCTION ex8(N NUMBER) RETURN ex13.tip\_ex8;  PROCEDURE ex9(denumireRestaurant restaurant.denumire%TYPE);  END ex13;  /    CREATE OR REPLACE PACKAGE BODY ex13 IS  PROCEDURE ex6 IS  TYPE tip\_tabel\_camera IS TABLE OF camera.cod\_camera%TYPE;  TYPE tip\_tabel\_zona\_de\_parcare IS TABLE OF zona\_de\_parcare.cod\_zona\_de\_parcare%TYPE INDEX BY BINARY\_INTEGER;  TYPE tip\_info\_hotel IS RECORD  (  cod\_hotel hotel.cod\_hotel%TYPE,  denumire hotel.denumire%TYPE,  tabel\_camera tip\_tabel\_camera,  tabel\_zona\_de\_parcare tip\_tabel\_zona\_de\_parcare  );  TYPE tip\_lista\_hotel IS VARRAY(100) OF tip\_info\_hotel;  tabel\_camera tip\_tabel\_camera;  tabel\_zona\_de\_parcare tip\_tabel\_zona\_de\_parcare;  lista\_hotel tip\_lista\_hotel := tip\_lista\_hotel();  TYPE tip\_contor IS RECORD  (  cod\_hotel hotel.cod\_hotel%TYPE,  denumire hotel.denumire%TYPE  );  v\_contor tip\_contor;  BEGIN  lista\_hotel.DELETE;  FOR v\_contor IN (SELECT cod\_hotel, denumire FROM hotel)  LOOP  SELECT cod\_camera BULK COLLECT INTO tabel\_camera  FROM camera WHERE camera.cod\_hotel = v\_contor.cod\_hotel;    SELECT cod\_zona\_de\_parcare BULK COLLECT INTO tabel\_zona\_de\_parcare  FROM zona\_de\_parcare WHERE zona\_de\_parcare.cod\_hotel = v\_contor.cod\_hotel;    lista\_hotel.EXTEND;  lista\_hotel(lista\_hotel.LAST) := tip\_info\_hotel(v\_contor.cod\_hotel, v\_contor.denumire, tabel\_camera, tabel\_zona\_de\_parcare);  END LOOP;    FOR i IN lista\_hotel.FIRST..lista\_hotel.LAST  LOOP  DBMS\_OUTPUT.PUT\_LINE('Hotelul ' || lista\_hotel(i).cod\_hotel || ', cu denumirea ' || lista\_hotel(i).denumire || ', are urmatoarele camere si zone de parcare:');    DBMS\_OUTPUT.PUT\_LINE('Camere:');  IF lista\_hotel(i).tabel\_camera.COUNT = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('nu exista camere');  ELSE  FOR j IN lista\_hotel(i).tabel\_camera.FIRST..lista\_hotel(i).tabel\_camera.LAST  LOOP  DBMS\_OUTPUT.PUT\_LINE('camera ' || j);  END LOOP;  END IF;    DBMS\_OUTPUT.PUT\_LINE('Zone de parcare:');  IF lista\_hotel(i).tabel\_zona\_de\_parcare.COUNT = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('nu exista zone de parcare');  ELSE  FOR j IN lista\_hotel(i).tabel\_zona\_de\_parcare.FIRST..lista\_hotel(i).tabel\_zona\_de\_parcare.LAST  LOOP  DBMS\_OUTPUT.PUT\_LINE('zona de parcare ' || j);  END LOOP;  END IF;    END LOOP;  END ex6;    PROCEDURE ex7 IS  CURSOR c\_locatie IS  SELECT l.cod\_locatie, l.adresa,  CURSOR(SELECT COUNT(\*) FROM restaurant r WHERE r.cod\_locatie = l.cod\_locatie)  FROM locatie l;    CURSOR c\_fel\_de\_mancare(codLocatie locatie.cod\_locatie%TYPE) IS  SELECT DISTINCT m.cod\_fel\_de\_mancare, m.denumire  FROM fel\_de\_mancare m JOIN restaurant\_serveste\_fel\_de\_mancare rm ON  m.cod\_fel\_de\_mancare = rm.cod\_fel\_de\_mancare  JOIN restaurant r ON rm.cod\_restaurant = r.cod\_restaurant  WHERE r.cod\_locatie = codLocatie;    v\_cod\_locatie locatie.cod\_locatie%TYPE;  v\_adresa locatie.adresa%TYPE;    TYPE tip\_ref\_cursor IS REF CURSOR;  v\_ref\_cursor tip\_ref\_cursor;    v\_numar\_restaurante NUMBER(10);    v\_cod\_fel\_de\_mancare fel\_de\_mancare.cod\_fel\_de\_mancare%TYPE;  v\_denumire fel\_de\_mancare.denumire%TYPE;    v\_numar\_feluri\_de\_mancare NUMBER(10);  BEGIN  OPEN c\_locatie;  LOOP  FETCH c\_locatie INTO v\_cod\_locatie, v\_adresa, v\_ref\_cursor;    EXIT WHEN c\_locatie%NOTFOUND;    v\_numar\_restaurante := 0;    LOOP  FETCH v\_ref\_cursor INTO v\_numar\_restaurante;    EXIT WHEN v\_ref\_cursor%NOTFOUND;  END LOOP;    IF v\_numar\_restaurante > 0 THEN  DBMS\_OUTPUT.PUT\_LINE('In locatia ' || v\_cod\_locatie || ', cu adresa ' || v\_adresa || ', se pot servi urmatoarele feluri de mancare');    v\_numar\_feluri\_de\_mancare := 0;    OPEN c\_fel\_de\_mancare(v\_cod\_locatie);  LOOP  FETCH c\_fel\_de\_mancare INTO v\_cod\_fel\_de\_mancare, v\_denumire;  EXIT WHEN c\_fel\_de\_mancare%NOTFOUND;    v\_numar\_feluri\_de\_mancare := v\_numar\_feluri\_de\_mancare + 1;    DBMS\_OUTPUT.PUT\_LINE(v\_cod\_fel\_de\_mancare || ' ' || v\_denumire);  END LOOP;  CLOSE c\_fel\_de\_mancare;    DBMS\_OUTPUT.PUT\_LINE('Numarul de feluri de mancare distincte este ' || v\_numar\_feluri\_de\_mancare);    END IF;    END LOOP;  CLOSE c\_locatie;  END ex7;    FUNCTION ex8(N NUMBER) RETURN ex13.tip\_ex8 IS  TYPE tip\_lista\_angajat IS VARRAY(100) OF ex13.tip\_ex8;  v\_aux\_angajat ex13.tip\_ex8;  v\_lista\_angajat tip\_lista\_angajat := tip\_lista\_angajat();  v\_lista\_aux\_angajat tip\_lista\_angajat := tip\_lista\_angajat();    v\_sortat BOOLEAN;  v\_deja\_luat BOOLEAN;    N\_INVALID EXCEPTION;  N\_PREA\_MARE EXCEPTION;    PRAGMA EXCEPTION\_INIT(N\_INVALID, -20001);  PRAGMA EXCEPTION\_INIT(N\_PREA\_MARE, -20002);    v\_nr\_angajati NUMBER(10) := 0;    BEGIN  SELECT COUNT(\*) INTO v\_nr\_angajati  FROM angajat;    IF v\_nr\_angajati = 0 THEN  RAISE NO\_DATA\_FOUND;  END IF;    IF N <= 0 THEN  RAISE N\_INVALID;  END IF;    IF v\_nr\_angajati < N THEN  RAISE N\_PREA\_MARE;  END IF;    --paznic  SELECT  p.cod\_angajat,  a.nume,  a.prenume,  a.salariu,  p.cod\_zona\_de\_parcare,  'paznic',  l.cod\_locatie,  l.adresa  BULK COLLECT INTO v\_lista\_aux\_angajat  FROM paznic p LEFT JOIN angajat a ON (p.cod\_angajat = a.cod\_angajat)  LEFT JOIN zona\_de\_parcare z ON (p.cod\_zona\_de\_parcare = z.cod\_zona\_de\_parcare)  LEFT JOIN locatie l ON (z.cod\_locatie = l.cod\_locatie);    FOR i IN v\_lista\_aux\_angajat.FIRST..v\_lista\_aux\_angajat.LAST  LOOP  v\_deja\_luat := FALSE;    IF v\_lista\_angajat.COUNT > 0 THEN  FOR j IN v\_lista\_angajat.FIRST..v\_lista\_angajat.LAST  LOOP  IF v\_lista\_aux\_angajat(i).cod\_angajat = v\_lista\_angajat(j).cod\_angajat THEN  v\_deja\_luat := TRUE;  END IF;  END LOOP;  END IF;    IF v\_deja\_luat = FALSE THEN  v\_lista\_angajat.EXTEND;  v\_lista\_angajat(v\_lista\_angajat.COUNT) := v\_lista\_aux\_angajat(i);  END IF;  END LOOP;  --receptionist  SELECT  r.cod\_angajat,  a.nume,  a.prenume,  a.salariu,  r.cod\_hotel,  'receptionist',  l.cod\_locatie,  l.adresa  BULK COLLECT INTO v\_lista\_aux\_angajat  FROM receptionist r LEFT JOIN angajat a ON (r.cod\_angajat = a.cod\_angajat)  LEFT JOIN hotel h ON (r.cod\_hotel = h.cod\_hotel)  LEFT JOIN locatie l ON (h.cod\_locatie = l.cod\_locatie);    FOR i IN v\_lista\_aux\_angajat.FIRST..v\_lista\_aux\_angajat.LAST  LOOP  v\_deja\_luat := FALSE;    IF v\_lista\_angajat.COUNT > 0 THEN  FOR j IN v\_lista\_angajat.FIRST..v\_lista\_angajat.LAST  LOOP  IF v\_lista\_aux\_angajat(i).cod\_angajat = v\_lista\_angajat(j).cod\_angajat THEN  v\_deja\_luat := TRUE;  END IF;  END LOOP;  END IF;    IF v\_deja\_luat = FALSE THEN  v\_lista\_angajat.EXTEND;  v\_lista\_angajat(v\_lista\_angajat.COUNT) := v\_lista\_aux\_angajat(i);  END IF;  END LOOP;  --administrator  SELECT  ad.cod\_angajat,  a.nume,  a.prenume,  a.salariu,  ad.cod\_hotel,  'administrator',  l.cod\_locatie,  l.adresa  BULK COLLECT INTO v\_lista\_aux\_angajat  FROM administrator ad LEFT JOIN angajat a ON (ad.cod\_angajat = a.cod\_angajat)  LEFT JOIN hotel h ON (ad.cod\_hotel = h.cod\_hotel)  LEFT JOIN locatie l ON (h.cod\_locatie = l.cod\_locatie);    FOR i IN v\_lista\_aux\_angajat.FIRST..v\_lista\_aux\_angajat.LAST  LOOP  v\_deja\_luat := FALSE;    IF v\_lista\_angajat.COUNT > 0 THEN  FOR j IN v\_lista\_angajat.FIRST..v\_lista\_angajat.LAST  LOOP  IF v\_lista\_aux\_angajat(i).cod\_angajat = v\_lista\_angajat(j).cod\_angajat THEN  v\_deja\_luat := TRUE;  END IF;  END LOOP;  END IF;    IF v\_deja\_luat = FALSE THEN  v\_lista\_angajat.EXTEND;  v\_lista\_angajat(v\_lista\_angajat.COUNT) := v\_lista\_aux\_angajat(i);  END IF;  END LOOP;  --chelner  SELECT  c.cod\_angajat,  a.nume,  a.prenume,  a.salariu,  c.cod\_restaurant,  'chelner',  l.cod\_locatie,  l.adresa  BULK COLLECT INTO v\_lista\_aux\_angajat  FROM chelner c LEFT JOIN angajat a ON (c.cod\_angajat = a.cod\_angajat)  LEFT JOIN restaurant r ON (c.cod\_restaurant = r.cod\_restaurant)  LEFT JOIN locatie l ON (r.cod\_locatie = l.cod\_locatie);    FOR i IN v\_lista\_aux\_angajat.FIRST..v\_lista\_aux\_angajat.LAST  LOOP  v\_deja\_luat := FALSE;    IF v\_lista\_angajat.COUNT > 0 THEN  FOR j IN v\_lista\_angajat.FIRST..v\_lista\_angajat.LAST  LOOP  IF v\_lista\_aux\_angajat(i).cod\_angajat = v\_lista\_angajat(j).cod\_angajat THEN  v\_deja\_luat := TRUE;  END IF;  END LOOP;  END IF;    IF v\_deja\_luat = FALSE THEN  v\_lista\_angajat.EXTEND;  v\_lista\_angajat(v\_lista\_angajat.COUNT) := v\_lista\_aux\_angajat(i);  END IF;  END LOOP;  --bucatar  SELECT  b.cod\_angajat,  a.nume,  a.prenume,  a.salariu,  b.cod\_restaurant,  'bucatar',  l.cod\_locatie,  l.adresa  BULK COLLECT INTO v\_lista\_aux\_angajat  FROM bucatar b LEFT JOIN angajat a ON (b.cod\_angajat = a.cod\_angajat)  LEFT JOIN restaurant r ON (b.cod\_restaurant = r.cod\_restaurant)  LEFT JOIN locatie l ON (r.cod\_locatie = l.cod\_locatie);    FOR i IN v\_lista\_aux\_angajat.FIRST..v\_lista\_aux\_angajat.LAST  LOOP  v\_deja\_luat := FALSE;    IF v\_lista\_angajat.COUNT > 0 THEN  FOR j IN v\_lista\_angajat.FIRST..v\_lista\_angajat.LAST  LOOP  IF v\_lista\_aux\_angajat(i).cod\_angajat = v\_lista\_angajat(j).cod\_angajat THEN  v\_deja\_luat := TRUE;  END IF;  END LOOP;  END IF;    IF v\_deja\_luat = FALSE THEN  v\_lista\_angajat.EXTEND;  v\_lista\_angajat(v\_lista\_angajat.COUNT) := v\_lista\_aux\_angajat(i);  END IF;  END LOOP;    --mai trebuie selectati exact o data angajatii care nu lucreaza pe niciun post din cele mentionate  --(paznic, receptionist, administrator, chelner, bucatar)  FOR i IN (SELECT cod\_angajat, nume, prenume, salariu FROM angajat)  LOOP  v\_deja\_luat := FALSE;    FOR j IN v\_lista\_angajat.FIRST..v\_lista\_angajat.LAST  LOOP  IF v\_lista\_angajat(j).cod\_angajat = i.cod\_angajat THEN  v\_deja\_luat := TRUE;  END IF;  END LOOP;    IF v\_deja\_luat = FALSE THEN  v\_lista\_angajat.EXTEND;  v\_lista\_angajat(v\_lista\_angajat.COUNT) := ex13.tip\_ex8(i.cod\_angajat, i.nume, i.prenume, i.salariu, null, '', null, null);  END IF;    END LOOP;    v\_sortat := FALSE;    WHILE v\_sortat = FALSE --BubbleSort, sortez dupa salariu lista tuturor angajatilor  LOOP  v\_sortat := TRUE;    FOR i IN 1..(v\_lista\_angajat.COUNT - 1)  LOOP  IF v\_lista\_angajat(i).salariu < v\_lista\_angajat(i + 1).salariu THEN  v\_sortat := FALSE;  v\_aux\_angajat := v\_lista\_angajat(i);  v\_lista\_angajat(i) := v\_lista\_angajat(i + 1);  v\_lista\_angajat(i + 1) := v\_aux\_angajat;  END IF;  END LOOP;    END LOOP;    RETURN v\_lista\_angajat(N);  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista angajati in baza de date');  RETURN NULL;  WHEN N\_INVALID THEN  DBMS\_OUTPUT.PUT\_LINE('Valoarea N data ca parametru functiei este un numar invalid (negativ sau 0)');  RETURN NULL;  WHEN N\_PREA\_MARE THEN  DBMS\_OUTPUT.PUT\_LINE('Valoarea N data ca parametru functiei este un numar mai mare decat numarul total de angajati');  RETURN NULL;  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Alte erori');  RETURN NULL;  END ex8;    PROCEDURE ex9(denumireRestaurant restaurant.denumire%TYPE) IS  TYPE tip\_info\_restaurant IS RECORD  (  cod\_restaurant restaurant.cod\_restaurant%TYPE,  numar\_stele restaurant.numar\_stele%TYPE,  adresa locatie.adresa%TYPE,  cod\_fel\_de\_mancare fel\_de\_mancare.cod\_fel\_de\_mancare%TYPE,  denumire\_fel\_de\_mancare fel\_de\_mancare.denumire%TYPE,  cod\_furnizor furnizor.cod\_furnizor%TYPE,  cod\_hotel hotel.cod\_hotel%TYPE,  denumire\_hotel hotel.denumire%TYPE  );  TYPE tip\_lista\_info\_restaurant IS VARRAY(1000) OF tip\_info\_restaurant;  v\_info\_restaurant tip\_info\_restaurant;  v\_lista\_info\_restaurant tip\_lista\_info\_restaurant := tip\_lista\_info\_restaurant();    v\_cod\_restaurant restaurant.cod\_restaurant%TYPE;    TYPE tip\_info\_fel\_de\_mancare IS RECORD  (  cod\_fel\_de\_mancare fel\_de\_mancare.cod\_fel\_de\_mancare%TYPE,  denumire fel\_de\_mancare.denumire%TYPE  );    TYPE tip\_tabel\_feluri\_de\_mancare IS TABLE OF tip\_info\_fel\_de\_mancare;  TYPE tip\_tabel\_furnizori IS TABLE OF furnizor.cod\_furnizor%TYPE;    tabel\_feluri\_de\_mancare tip\_tabel\_feluri\_de\_mancare := tip\_tabel\_feluri\_de\_mancare();  tabel\_furnizori tip\_tabel\_furnizori := tip\_tabel\_furnizori();    v\_deja\_exista BOOLEAN := FALSE;    CURSOR c\_info\_restaurant(codRestaurant restaurant.cod\_restaurant%TYPE)  IS  SELECT r.cod\_restaurant, r.numar\_stele, l.adresa, m.cod\_fel\_de\_mancare, m.denumire, f.cod\_furnizor, h.cod\_hotel, h.denumire  FROM restaurant r LEFT JOIN locatie l ON (r.cod\_locatie = l.cod\_locatie)  LEFT JOIN restaurant\_serveste\_fel\_de\_mancare rm ON (r.cod\_restaurant = rm.cod\_restaurant)  LEFT JOIN fel\_de\_mancare m ON (rm.cod\_fel\_de\_mancare = m.cod\_fel\_de\_mancare)  LEFT JOIN hotel h ON (r.cod\_hotel = h.cod\_hotel)  LEFT JOIN furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte frc ON (r.cod\_restaurant = frc.cod\_restaurant)  LEFT JOIN furnizor f ON (frc.cod\_furnizor = f.cod\_furnizor)  WHERE r.cod\_restaurant = codRestaurant;    NICIUN\_FEL\_DE\_MANCARE EXCEPTION;  NICIUN\_FURNIZOR EXCEPTION;    PRAGMA EXCEPTION\_INIT(NICIUN\_FEL\_DE\_MANCARE, -20001);  PRAGMA EXCEPTION\_INIT(NICIUN\_FURNIZOR, -20002);    BEGIN  SELECT cod\_restaurant INTO v\_cod\_restaurant --acest query va declansa exceptiile NO\_DATA\_FOUND sau TOO\_MANY\_ROWS  FROM restaurant WHERE INITCAP(denumire) = INITCAP(denumireRestaurant);    OPEN c\_info\_restaurant(v\_cod\_restaurant);    FETCH c\_info\_restaurant INTO v\_info\_restaurant;    WHILE c\_info\_restaurant%FOUND  LOOP  v\_lista\_info\_restaurant.EXTEND;  v\_lista\_info\_restaurant(v\_lista\_info\_restaurant.COUNT) := v\_info\_restaurant;  FETCH c\_info\_restaurant INTO v\_info\_restaurant;  END LOOP;    CLOSE c\_info\_restaurant;      FOR i IN v\_lista\_info\_restaurant.FIRST..v\_lista\_info\_restaurant.LAST  LOOP  IF v\_lista\_info\_restaurant(i).cod\_fel\_de\_mancare IS NOT NULL THEN    v\_deja\_exista := FALSE;    IF tabel\_feluri\_de\_mancare.COUNT > 0 THEN  FOR j IN tabel\_feluri\_de\_mancare.FIRST..tabel\_feluri\_de\_mancare.LAST  LOOP  IF tabel\_feluri\_de\_mancare(j).cod\_fel\_de\_mancare = v\_lista\_info\_restaurant(i).cod\_fel\_de\_mancare THEN  v\_deja\_exista := TRUE;  END IF;  END LOOP;  END IF;    IF v\_deja\_exista = FALSE THEN  tabel\_feluri\_de\_mancare.EXTEND;  tabel\_feluri\_de\_mancare(tabel\_feluri\_de\_mancare.LAST) := tip\_info\_fel\_de\_mancare(v\_lista\_info\_restaurant(i).cod\_fel\_de\_mancare, v\_lista\_info\_restaurant(i).denumire\_fel\_de\_mancare);  END IF;  END IF;    IF v\_lista\_info\_restaurant(i).cod\_furnizor IS NOT NULL THEN  IF v\_lista\_info\_restaurant(i).cod\_furnizor NOT MEMBER OF tabel\_furnizori THEN  tabel\_furnizori.EXTEND;  tabel\_furnizori(tabel\_furnizori.LAST) := v\_lista\_info\_restaurant(i).cod\_furnizor;  END IF;  END IF;  END LOOP;    DBMS\_OUTPUT.PUT\_LINE('Restaurantul ' || v\_lista\_info\_restaurant(1).cod\_restaurant || ' ' || denumireRestaurant  || ' cu numarul de stele ' || v\_lista\_info\_restaurant(1).numar\_stele  || ' aflat la adresa ' || v\_lista\_info\_restaurant(1).adresa || ', ce apartine de hotelul '  || v\_lista\_info\_restaurant(1).cod\_hotel || ' ' || v\_lista\_info\_restaurant(1).denumire\_hotel  || ', serveste urmatoarele feluri de mancare: ');      IF tabel\_feluri\_de\_mancare.COUNT > 0 THEN  FOR i IN tabel\_feluri\_de\_mancare.FIRST..tabel\_feluri\_de\_mancare.LAST  LOOP  DBMS\_OUTPUT.PUT\_LINE(tabel\_feluri\_de\_mancare(i).cod\_fel\_de\_mancare || ' ' || tabel\_feluri\_de\_mancare(i).denumire);  END LOOP;  ELSE  RAISE NICIUN\_FEL\_DE\_MANCARE;  END IF;    DBMS\_OUTPUT.PUT\_LINE('Si are contracte semnate cu urmatorii furnizori: ');    IF tabel\_furnizori.COUNT > 0 THEN  FOR i IN tabel\_furnizori.FIRST..tabel\_furnizori.LAST  LOOP  DBMS\_OUTPUT.PUT\_LINE(tabel\_furnizori(i));  END LOOP;  ELSE  RAISE NICIUN\_FURNIZOR;  END IF;    FORALL i IN tabel\_furnizori.FIRST..tabel\_furnizori.LAST  DELETE FROM furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte frc  WHERE frc.cod\_furnizor = tabel\_furnizori(i) AND frc.cod\_restaurant = v\_cod\_restaurant;    EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Restaurantul cu denumirea ' || denumireRestaurant || ' nu apare in baza de date');  WHEN TOO\_MANY\_ROWS THEN  DBMS\_OUTPUT.PUT\_LINE('Restaurantul cu denumirea ' || denumireRestaurant || ' apare de mai multe ori in baza de date');  WHEN NICIUN\_FEL\_DE\_MANCARE THEN  DBMS\_OUTPUT.PUT\_LINE('Restaurantul cu denumirea ' || denumireRestaurant || ' nu ofera niciun fel de mancare');  WHEN NICIUN\_FURNIZOR THEN  DBMS\_OUTPUT.PUT\_LINE('Restaurantul cu denumirea ' || denumireRestaurant || ' nu are contract incheiat cu niciun furnizor');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Alte erori');  END ex9;  END ex13;  /    BEGIN  ex13.ex6;  ex13.ex7;  END;  /    --pentru ex8    DECLARE  --v\_n NUMBER(10) := &p\_n;  v\_rezultat ex13.tip\_ex8;  BEGIN  v\_rezultat := ex13.ex8(-1);  DBMS\_OUTPUT.PUT\_LINE('Angajatul ' || v\_rezultat.cod\_angajat || ' ' || v\_rezultat.nume || ' '  || v\_rezultat.prenume || ', avand salariul ' || v\_rezultat.salariu || ', lucreaza la ' || v\_rezultat.cod\_munca  || ', pe post de ' || v\_rezultat.tip\_munca || ', locatia muncii fiind ' || v\_rezultat.cod\_locatie || ' ' ||  v\_rezultat.adresa);    v\_rezultat := ex13.ex8(0);  DBMS\_OUTPUT.PUT\_LINE('Angajatul ' || v\_rezultat.cod\_angajat || ' ' || v\_rezultat.nume || ' '  || v\_rezultat.prenume || ', avand salariul ' || v\_rezultat.salariu || ', lucreaza la ' || v\_rezultat.cod\_munca  || ', pe post de ' || v\_rezultat.tip\_munca || ', locatia muncii fiind ' || v\_rezultat.cod\_locatie || ' ' ||  v\_rezultat.adresa);    v\_rezultat := ex13.ex8(4);  DBMS\_OUTPUT.PUT\_LINE('Angajatul ' || v\_rezultat.cod\_angajat || ' ' || v\_rezultat.nume || ' '  || v\_rezultat.prenume || ', avand salariul ' || v\_rezultat.salariu || ', lucreaza la ' || v\_rezultat.cod\_munca  || ', pe post de ' || v\_rezultat.tip\_munca || ', locatia muncii fiind ' || v\_rezultat.cod\_locatie || ' ' ||  v\_rezultat.adresa);    v\_rezultat := ex13.ex8(101);  DBMS\_OUTPUT.PUT\_LINE('Angajatul ' || v\_rezultat.cod\_angajat || ' ' || v\_rezultat.nume || ' '  || v\_rezultat.prenume || ', avand salariul ' || v\_rezultat.salariu || ', lucreaza la ' || v\_rezultat.cod\_munca  || ', pe post de ' || v\_rezultat.tip\_munca || ', locatia muncii fiind ' || v\_rezultat.cod\_locatie || ' ' ||  v\_rezultat.adresa);  END;  /    ROLLBACK;  COMMIT;    DELETE FROM paznic;  DELETE FROM receptionist;  DELETE FROM administrator;  DELETE FROM chelner;  DELETE FROM bucatar;    DELETE FROM angajat;    --pentru ex9    INSERT INTO RESTAURANT(cod\_restaurant, cod\_hotel, cod\_locatie, denumire, numar\_stele, an\_infiintare)  VALUES(SECVENTA\_RESTAURANT.NEXTVAL, 4, 7, 'Madrid', NULL, 1993); -- restaurantul 'Madrid' va declansa exceptia de TOO\_MANY\_ROWS  INSERT INTO RESTAURANT(cod\_restaurant, cod\_hotel, cod\_locatie, denumire, numar\_stele, an\_infiintare)  VALUES(SECVENTA\_RESTAURANT.NEXTVAL, 4, 7, 'Londra3', NULL, 1973); -- restaurantul 'Londra3' va declansa exceptia NICIUN\_FEL\_DE\_MANCARE    DECLARE  --v\_denumire\_restaurant restaurant.denumire%TYPE := Initcap('&p\_denumire\_restaurant');  BEGIN  ex13.ex9('Londra3'); --nu are niciun fel de mancare  ex13.ex9('Madrid'); --too\_many\_rows  ex13.ex9('Paris'); --nu are niciun fel de mancare  ex13.ex9('Napoli'); --nu are furnizori  ex13.ex9('Amsterdam'); --nu exista  ex13.ex9('Roma'); --ok  END;  /    ROLLBACK;  COMMIT;    SELECT \* FROM restaurant;  SELECT \* FROM furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte;  SELECT \* FROM restaurant\_serveste\_fel\_de\_mancare;    ALTER TRIGGER ex10 DISABLE;  ALTER TRIGGER ex10 ENABLE; |

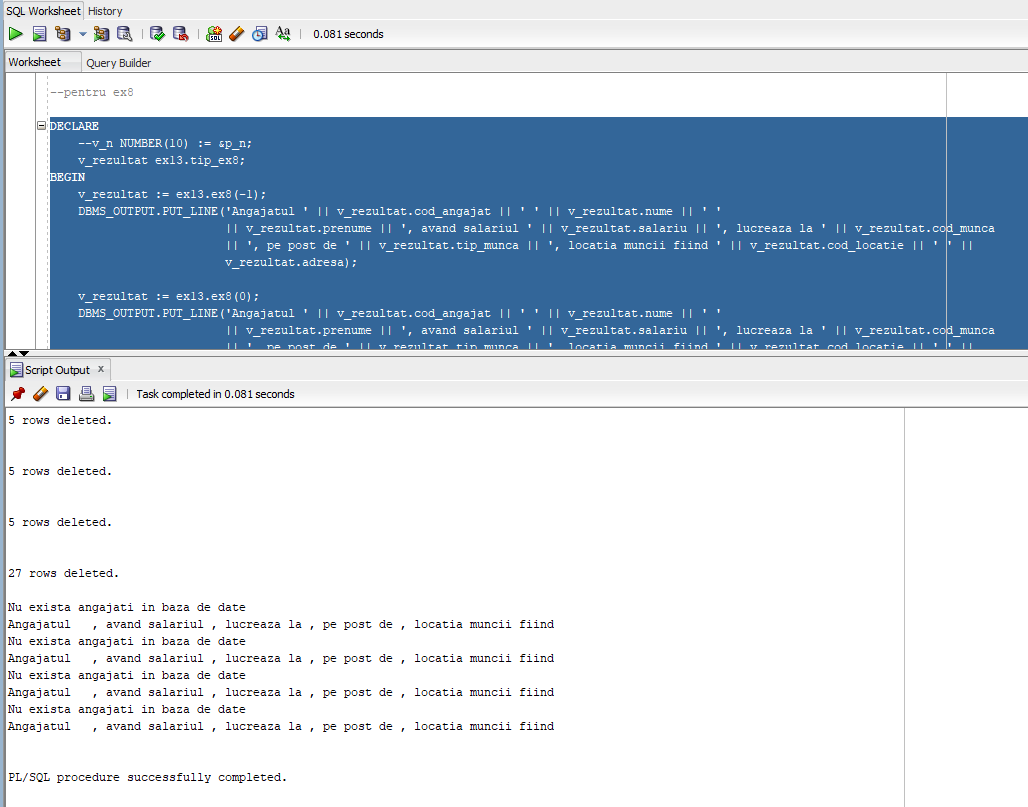


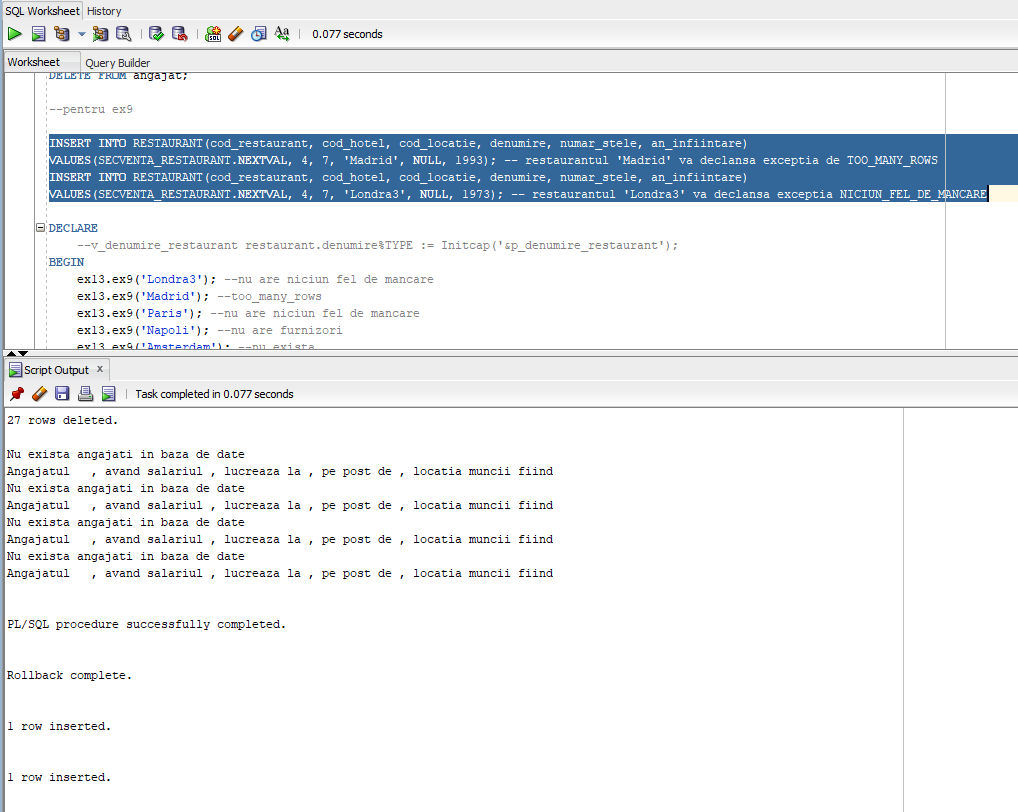


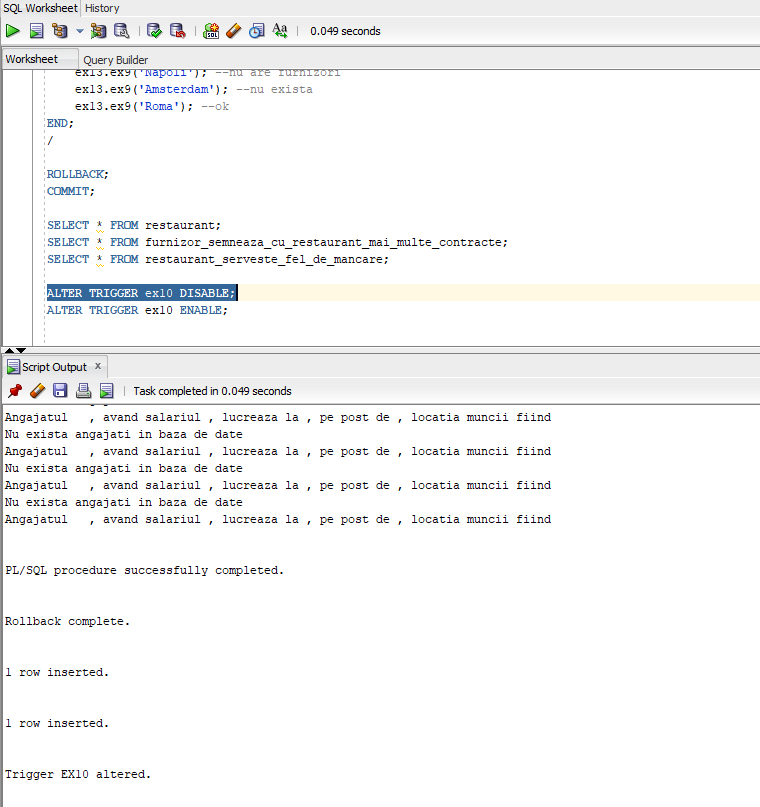


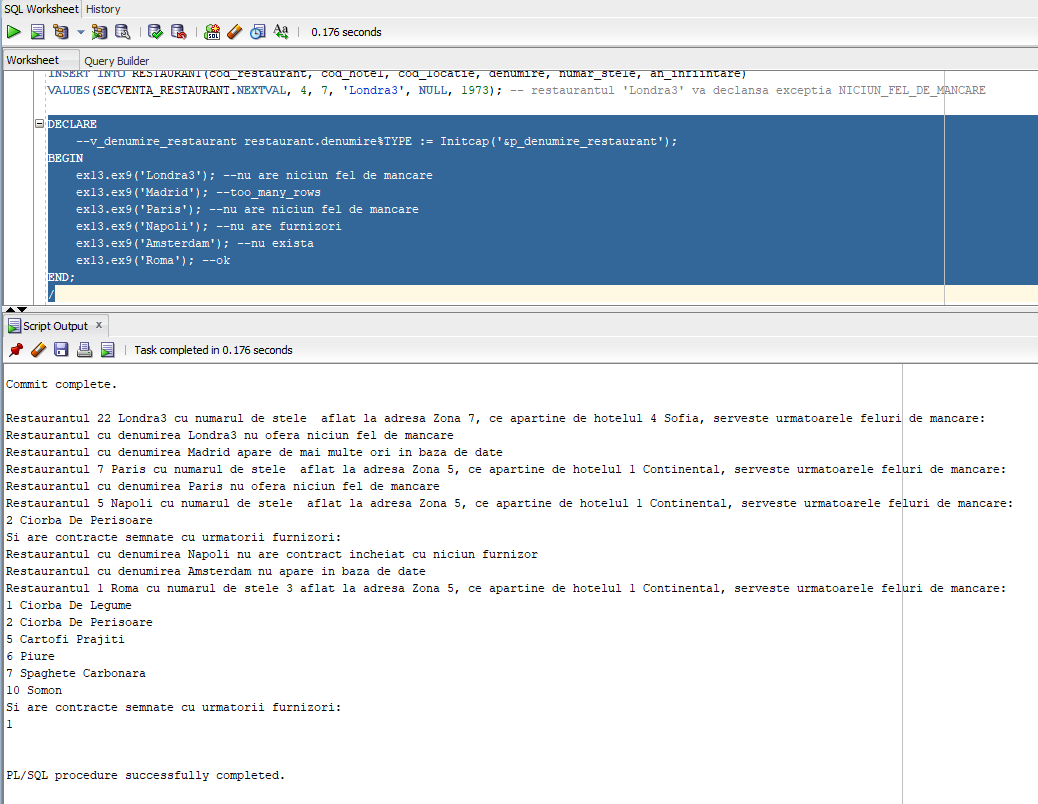












# Exercitiul 14

Definiti un pachet care sa includa tipuri de date complexe si obiecte necesare unui flux de actiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 functii, minim 2 proceduri).

|  |
| --- |
| /\*  Exercitiul 14  Pachet    Procedura adauga\_facilitate\_camera primeste o camera si o facilitate si asociaza facilitatea respectiva camerei.  Procedura salariu\_mediu\_angajat intoarce in parametrul de out salariul mediu al tuturor angajatilor.  Procedura afisare\_sortata\_hotel afiseaza hotelurile in ordinea crescatoare a numarului de camere din fiecare hotel.    Functia angajati\_dupa\_salariu intoarce un tabel ce contine angajatii care au salariul sub o valoare data ca parametru.  Functia cel\_mai\_lung\_contract\_restaurant intoarce informatii despre cel mai longeviv contract pe care restaurantul dat ca parametru l-a avut.  Functia creste\_salariu\_angajat creste salariul cu x% al angajatilor ce inainte aveau salariul intr-un interval de valori dat ca parametru. Functia intoarce cati angajati au beneficiat de crestere.  \*/    ----------------------------------------------------------    SET SERVEROUTPUT ON  SET VERIFY OFF    CREATE OR REPLACE PACKAGE ex14 IS  TYPE tip\_hotel IS RECORD  (  cod\_hotel hotel.cod\_hotel%TYPE,  denumire hotel.denumire%TYPE,  nr\_camere NUMBER(10)  );  TYPE tip\_tabel\_hotel IS TABLE OF tip\_hotel INDEX BY BINARY\_INTEGER;    TYPE tip\_angajat IS RECORD  (  cod\_angajat angajat.cod\_angajat%TYPE,  nume angajat.nume%TYPE,  prenume angajat.prenume%TYPE,  salariu angajat.salariu%TYPE  );  TYPE tip\_tabel\_angajat IS TABLE OF tip\_angajat;    TYPE tip\_contract IS RECORD  (  cod\_contract contract.cod\_contract%TYPE,  data\_inceput contract.data\_inceput%TYPE,  data\_sfarsit contract.data\_sfarsit%TYPE,  cod\_furnizor furnizor.cod\_furnizor%TYPE,  nume\_furnizor furnizor.nume%TYPE,  durata\_contract\_in\_zile NUMBER(10)  );    PROCEDURE adauga\_facilitate\_camera(p\_cod\_camera IN NUMBER, p\_cod\_facilitate IN NUMBER);  PROCEDURE salariu\_mediu\_angajat(p\_salariu\_mediu OUT NUMBER);  PROCEDURE afisare\_sortata\_hotel;    FUNCTION angajati\_dupa\_salariu(p\_salariu NUMBER) RETURN tip\_tabel\_angajat;  FUNCTION cel\_mai\_lung\_contract\_restaurant(p\_cod\_restaurant NUMBER) RETURN tip\_contract;  FUNCTION creste\_salariu\_angajat(p\_x NUMBER, p\_salariu\_minim NUMBER, p\_salariu\_maxim NUMBER) RETURN NUMBER;  END ex14;  /    CREATE OR REPLACE PACKAGE BODY ex14 IS  PROCEDURE adauga\_facilitate\_camera(p\_cod\_camera IN NUMBER, p\_cod\_facilitate IN NUMBER) IS  NU\_EXISTA\_CAMERA EXCEPTION;  NU\_EXISTA\_FACILITATE EXCEPTION;  FACILITATE\_DEJA\_EXISTENTA EXCEPTION;    PRAGMA EXCEPTION\_INIT(NU\_EXISTA\_CAMERA, -20001);  PRAGMA EXCEPTION\_INIT(NU\_EXISTA\_FACILITATE, -20002);  PRAGMA EXCEPTION\_INIT(FACILITATE\_DEJA\_EXISTENTA, -20003);    v\_nr NUMBER(10);  v\_cod\_hotel NUMBER(10);  BEGIN  SELECT COUNT(\*) INTO v\_nr FROM camera  WHERE cod\_camera = p\_cod\_camera;    IF v\_nr = 0 THEN  RAISE NU\_EXISTA\_CAMERA;  END IF;    SELECT COUNT(\*) INTO v\_nr FROM facilitate  WHERE cod\_facilitate = p\_cod\_facilitate;    IF v\_nr = 0 THEN  RAISE NU\_EXISTA\_FACILITATE;  END IF;    SELECT COUNT(\*) INTO v\_nr FROM camera\_prezinta\_facilitate  WHERE cod\_camera = p\_cod\_camera AND cod\_facilitate = p\_cod\_facilitate;    IF v\_nr > 0 THEN  RAISE FACILITATE\_DEJA\_EXISTENTA;  END IF;    SELECT cod\_hotel INTO v\_cod\_hotel FROM camera  WHERE cod\_camera = p\_cod\_camera;    INSERT INTO camera\_prezinta\_facilitate(cod\_camera, cod\_hotel, cod\_facilitate)  VALUES (p\_cod\_camera, v\_cod\_hotel, p\_cod\_facilitate);    EXCEPTION  WHEN NU\_EXISTA\_CAMERA THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista camera cu id-ul ' || p\_cod\_camera);  WHEN NU\_EXISTA\_FACILITATE THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista facilitatea cu id-ul ' || p\_cod\_facilitate);  WHEN FACILITATE\_DEJA\_EXISTENTA THEN  DBMS\_OUTPUT.PUT\_LINE('Facilitatea cu id-ul ' || p\_cod\_facilitate || ' deja este asociata camerei cu id-ul ' || p\_cod\_camera);  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Alte erori');  END adauga\_facilitate\_camera;    PROCEDURE salariu\_mediu\_angajat(p\_salariu\_mediu OUT NUMBER) IS  NU\_EXISTA\_ANGAJATI EXCEPTION;    PRAGMA EXCEPTION\_INIT(NU\_EXISTA\_ANGAJATI, -20001);    v\_nr NUMBER(10);  BEGIN  SELECT COUNT(\*) INTO v\_nr FROM angajat;    IF v\_nr = 0 THEN  RAISE NU\_EXISTA\_ANGAJATI;  END IF;    SELECT AVG(salariu) INTO p\_salariu\_mediu FROM angajat;    EXCEPTION  WHEN NU\_EXISTA\_ANGAJATI THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista angajati');  p\_salariu\_mediu := -20001;  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Alte erori');  p\_salariu\_mediu := -1;  END salariu\_mediu\_angajat;    PROCEDURE afisare\_sortata\_hotel IS  tabel\_hotel tip\_tabel\_hotel;    NU\_EXISTA\_HOTELURI EXCEPTION;    PRAGMA EXCEPTION\_INIT(NU\_EXISTA\_HOTELURI, -20001);    v\_nr NUMBER(10);    v\_nr\_elemente NUMBER(10) := 0;    v\_sortat BOOLEAN;    v\_auxiliar tip\_hotel;  BEGIN  SELECT COUNT(\*) INTO v\_nr FROM hotel;    IF v\_nr = 0 THEN  RAISE NU\_EXISTA\_HOTELURI;  END IF;    FOR x IN (SELECT cod\_hotel, denumire FROM hotel)  LOOP  v\_nr\_elemente := v\_nr\_elemente + 1;  tabel\_hotel(v\_nr\_elemente) := tip\_hotel(x.cod\_hotel, x.denumire, 0);  FOR y IN (SELECT cod\_hotel FROM camera WHERE cod\_hotel = x.cod\_hotel)  LOOP  tabel\_hotel(v\_nr\_elemente).nr\_camere := tabel\_hotel(v\_nr\_elemente).nr\_camere + 1;  END LOOP;  END LOOP;    --sortare (bubblesort)  v\_sortat := FALSE;  WHILE v\_sortat = FALSE  LOOP  v\_sortat := TRUE;    FOR x IN 2..v\_nr\_elemente  LOOP  IF tabel\_hotel(x).nr\_camere < tabel\_hotel(x - 1).nr\_camere THEN  v\_sortat := FALSE;    v\_auxiliar := tabel\_hotel(x - 1);  tabel\_hotel(x - 1) := tabel\_hotel(x);  tabel\_hotel(x) := v\_auxiliar;    END IF;  END LOOP;    END LOOP;    --afisare  FOR x IN 1..v\_nr\_elemente  LOOP  DBMS\_OUTPUT.PUT\_LINE(tabel\_hotel(x).cod\_hotel || ' ' || tabel\_hotel(x).denumire || ' ' || tabel\_hotel(x).nr\_camere);  END LOOP;    EXCEPTION  WHEN NU\_EXISTA\_HOTELURI THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista hoteluri in baza de date');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Alte erori');  END afisare\_sortata\_hotel;    FUNCTION angajati\_dupa\_salariu(p\_salariu NUMBER) RETURN tip\_tabel\_angajat IS  NU\_EXISTA\_ANGAJATI EXCEPTION;    PRAGMA EXCEPTION\_INIT(NU\_EXISTA\_ANGAJATI, -20001);    v\_nr NUMBER(10);    v\_tabel\_angajat tip\_tabel\_angajat := tip\_tabel\_angajat();  BEGIN  SELECT COUNT(\*) INTO v\_nr FROM angajat WHERE salariu < p\_salariu;    IF v\_nr = 0 THEN  RAISE NU\_EXISTA\_ANGAJATI;  END IF;    SELECT cod\_angajat, nume, prenume, salariu BULK COLLECT INTO v\_tabel\_angajat  FROM angajat WHERE salariu < p\_salariu;    RETURN v\_tabel\_angajat;  EXCEPTION  WHEN NU\_EXISTA\_ANGAJATI THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista angajati cu salariul sub valoarea data');  RETURN v\_tabel\_angajat;  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Alte erori');  RETURN v\_tabel\_angajat;  END angajati\_dupa\_salariu;    FUNCTION cel\_mai\_lung\_contract\_restaurant(p\_cod\_restaurant NUMBER) RETURN tip\_contract IS  NU\_EXISTA\_RESTAURANT EXCEPTION;  NU\_EXISTA\_CONTRACT EXCEPTION;    PRAGMA EXCEPTION\_INIT(NU\_EXISTA\_RESTAURANT, -20001);  PRAGMA EXCEPTION\_INIT(NU\_EXISTA\_CONTRACT, -20002);    v\_contract tip\_contract := tip\_contract();    v\_nr NUMBER(10);    v\_nr\_zile NUMBER(10) := -1;    v\_nr\_zile\_crt NUMBER(10);    BEGIN  SELECT COUNT(\*) INTO v\_nr FROM restaurant  WHERE cod\_restaurant = p\_cod\_restaurant;    IF v\_nr = 0 THEN  RAISE NU\_EXISTA\_RESTAURANT;  END IF;    SELECT COUNT(\*) INTO v\_nr FROM furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte  WHERE cod\_restaurant = p\_cod\_restaurant;    IF v\_nr = 0 THEN  RAISE NU\_EXISTA\_CONTRACT;  END IF;    FOR x IN (SELECT c.cod\_contract, c.data\_inceput, c.data\_sfarsit, f.cod\_furnizor, f.nume  FROM contract c JOIN furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte frc ON (c.cod\_contract = frc.cod\_contract)  JOIN furnizor f ON (frc.cod\_furnizor = f.cod\_furnizor) JOIN restaurant r ON (frc.cod\_restaurant = r.cod\_restaurant)  WHERE r.cod\_restaurant = p\_cod\_restaurant)  LOOP  v\_nr\_zile\_crt := x.data\_sfarsit - x.data\_inceput;  IF v\_nr\_zile\_crt > v\_nr\_zile THEN  v\_nr\_zile := v\_nr\_zile\_crt;  v\_contract := tip\_contract(x.cod\_contract, x.data\_inceput, x.data\_sfarsit, x.cod\_furnizor, x.nume, v\_nr\_zile);  END IF;  END LOOP;    RETURN v\_contract;  EXCEPTION  WHEN NU\_EXISTA\_RESTAURANT THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista restaurantul cu id-ul ' || p\_cod\_restaurant);  RETURN v\_contract;  WHEN NU\_EXISTA\_CONTRACT THEN  DBMS\_OUTPUT.PUT\_LINE('Restaurantul cu id-ul ' || p\_cod\_restaurant || ' nu are niciun contract semnat');  RETURN v\_contract;  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Alte erori');  RETURN v\_contract;  END cel\_mai\_lung\_contract\_restaurant;    FUNCTION creste\_salariu\_angajat(p\_x NUMBER, p\_salariu\_minim NUMBER, p\_salariu\_maxim NUMBER) RETURN NUMBER IS  NICIUN\_ANGAJAT\_IN\_INTERVAL EXCEPTION;  INTERVAL\_INVALID EXCEPTION;  PROCENT\_INVALID EXCEPTION;    PRAGMA EXCEPTION\_INIT(NICIUN\_ANGAJAT\_IN\_INTERVAL, -20001);  PRAGMA EXCEPTION\_INIT(INTERVAL\_INVALID, -20002);  PRAGMA EXCEPTION\_INIT(PROCENT\_INVALID, -20003);    v\_nr NUMBER(10);  BEGIN  IF p\_x <= 0 OR p\_x > 1 THEN  RAISE PROCENT\_INVALID;  END IF;    IF p\_salariu\_minim > p\_salariu\_maxim THEN  RAISE INTERVAL\_INVALID;  END IF;    SELECT COUNT(\*) INTO v\_nr FROM angajat  WHERE p\_salariu\_minim <= salariu AND salariu <= p\_salariu\_maxim;    IF v\_nr = 0 THEN  RAISE NICIUN\_ANGAJAT\_IN\_INTERVAL;  END IF;    UPDATE angajat  SET salariu = salariu + salariu \* p\_x  WHERE p\_salariu\_minim <= salariu AND salariu <= p\_salariu\_maxim;    v\_nr := SQL%ROWCOUNT;    RETURN v\_nr;  EXCEPTION  WHEN NICIUN\_ANGAJAT\_IN\_INTERVAL THEN  DBMS\_OUTPUT.PUT\_LINE('Niciun angajat nu a fost afectat de modificare');  RETURN -20001;  WHEN INTERVAL\_INVALID THEN  DBMS\_OUTPUT.PUT\_LINE('Intervalul dat ca parametru este invalid');  RETURN -20002;  WHEN PROCENT\_INVALID THEN  DBMS\_OUTPUT.PUT\_LINE('Procentul de modificare este invalid');  RETURN -20003;  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Alte erori');  RETURN -20004;  END creste\_salariu\_angajat;    END ex14;  /    BEGIN  ex14.adauga\_facilitate\_camera(-1, 2); --exceptie (nu exista camera)  ex14.adauga\_facilitate\_camera(1, -2); --exceptie (nu exista facilitatea)  ex14.adauga\_facilitate\_camera(1, 2);  ex14.adauga\_facilitate\_camera(1, 2); --facilitatea deja exista pentru camera (exceptie)  END;  /    DELETE FROM paznic;  DELETE FROM receptionist;  DELETE FROM administrator;  DELETE FROM chelner;  DELETE FROM bucatar;    DELETE FROM angajat;    DECLARE  v\_salariu\_mediu NUMBER(10, 2);  BEGIN  ex14.salariu\_mediu\_angajat(v\_salariu\_mediu);  DBMS\_OUTPUT.PUT\_LINE(v\_salariu\_mediu);  END;  /    ROLLBACK;  COMMIT;    ALTER TRIGGER ex10 ENABLE;  ALTER TRIGGER ex10 DISABLE;    DELETE FROM bucatar;  DELETE FROM chelner;  DELETE FROM administrator;  DELETE FROM receptionist;  DELETE FROM paznic;  DELETE FROM angajat;  DELETE FROM furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte;  DELETE FROM restaurant\_serveste\_fel\_de\_mancare;  DELETE FROM restaurant;  DELETE FROM fel\_de\_mancare;  DELETE FROM contract;  DELETE FROM furnizor;  DELETE FROM loc\_de\_parcare;  DELETE FROM zona\_de\_parcare;  DELETE FROM client\_rezerva\_camera;  DELETE FROM camera\_prezinta\_facilitate;  DELETE FROM client;  DELETE FROM facilitate;  DELETE FROM camera;  DELETE FROM hotel;    BEGIN  ex14.afisare\_sortata\_hotel();  END;  /    DECLARE  tabel\_angajat ex14.tip\_tabel\_angajat;  BEGIN  tabel\_angajat := ex14.angajati\_dupa\_salariu(5000);  IF tabel\_angajat.COUNT > 0 THEN  FOR x IN tabel\_angajat.FIRST..tabel\_angajat.LAST  LOOP  DBMS\_OUTPUT.PUT\_LINE(tabel\_angajat(x).cod\_angajat || ' ' || tabel\_angajat(x).nume || ' ' || tabel\_angajat(x).prenume || ' ' || tabel\_angajat(x).salariu);  END LOOP;  END IF;    tabel\_angajat := ex14.angajati\_dupa\_salariu(1000); --exceptie, nu exista angajati  IF tabel\_angajat.COUNT > 0 THEN  FOR x IN tabel\_angajat.FIRST..tabel\_angajat.LAST  LOOP  DBMS\_OUTPUT.PUT\_LINE(tabel\_angajat(x).cod\_angajat || ' ' || tabel\_angajat(x).nume || ' ' || tabel\_angajat(x).prenume || ' ' || tabel\_angajat(x).salariu);  END LOOP;  END IF;  END;  /    DECLARE  contract ex14.tip\_contract;  BEGIN  contract := ex14.cel\_mai\_lung\_contract\_restaurant(1);  DBMS\_OUTPUT.PUT\_LINE(contract.cod\_contract || ' ' || contract.data\_inceput || ' ' || contract.data\_sfarsit || ' ' || contract.cod\_furnizor || ' ' || contract.nume\_furnizor || ' ' || contract.durata\_contract\_in\_zile);    contract := ex14.cel\_mai\_lung\_contract\_restaurant(-1); --nu exista restaurant  DBMS\_OUTPUT.PUT\_LINE(contract.cod\_contract || ' ' || contract.data\_inceput || ' ' || contract.data\_sfarsit || ' ' || contract.cod\_furnizor || ' ' || contract.nume\_furnizor || ' ' || contract.durata\_contract\_in\_zile);    contract := ex14.cel\_mai\_lung\_contract\_restaurant(5); --nu exista contracte pentru restaurant  DBMS\_OUTPUT.PUT\_LINE(contract.cod\_contract || ' ' || contract.data\_inceput || ' ' || contract.data\_sfarsit || ' ' || contract.cod\_furnizor || ' ' || contract.nume\_furnizor || ' ' || contract.durata\_contract\_in\_zile);  END;  /    DECLARE  v\_nr NUMBER(10);  BEGIN    v\_nr := ex14.creste\_salariu\_angajat(-0.01, 4510, 4520); --procent invalid  DBMS\_OUTPUT.PUT\_LINE(v\_nr);    v\_nr := ex14.creste\_salariu\_angajat(1.01, 4510, 4520); --procent invalid  DBMS\_OUTPUT.PUT\_LINE(v\_nr);    v\_nr := ex14.creste\_salariu\_angajat(0.1, 3000, 2000); --interval invalid  DBMS\_OUTPUT.PUT\_LINE(v\_nr);    v\_nr := ex14.creste\_salariu\_angajat(0.1, 2000, 3000); --niciun angajat in interval  DBMS\_OUTPUT.PUT\_LINE(v\_nr);    v\_nr := ex14.creste\_salariu\_angajat(0.1, 3900, 4100);  DBMS\_OUTPUT.PUT\_LINE(v\_nr);  END;  /    SELECT \* FROM angajat  ORDER BY salariu;    SELECT \* FROM angajat;    SELECT \* FROM camera\_prezinta\_facilitate;    SELECT \* FROM restaurant;    SELECT \* FROM furnizor\_semneaza\_cu\_restaurant\_mai\_multe\_contracte;    DELETE FROM camera\_prezinta\_facilitate  WHERE cod\_camera = 1 AND cod\_facilitate = 2;    COMMIT;  ROLLBACK; |

