**Sisteme de gestiune a bazelor de date**

Cofetărie

Prof. Curs: Gabriela Mihai

Prof. laborator: Anamaria Ștefania Marchidon (Zanoaga)

Proiect realizat de: Țelicov Letiția-Ioana

Grupa 232

Cuprins

[Descriere model 3](file:///C:\Users\letit\OneDrive\Desktop\facultate\132_Telicov_Letitia-Ioana-proiect%20(1).docx#_Toc135927140)

[Diagrama entitate-relație 3](file:///C:\Users\letit\OneDrive\Desktop\facultate\132_Telicov_Letitia-Ioana-proiect%20(1).docx#_Toc135927141)

[Diagrama conceptuală 4](file:///C:\Users\letit\OneDrive\Desktop\facultate\132_Telicov_Letitia-Ioana-proiect%20(1).docx#_Toc135927142)

[Descriere ENTITĂȚI 5](file:///C:\Users\letit\OneDrive\Desktop\facultate\132_Telicov_Letitia-Ioana-proiect%20(1).docx#_Toc135927143)

[DESCRIERE RELAȚII 6](file:///C:\Users\letit\OneDrive\Desktop\facultate\132_Telicov_Letitia-Ioana-proiect%20(1).docx#_Toc135927144)

[DESCRIERE ATRIBUTE 9](file:///C:\Users\letit\OneDrive\Desktop\facultate\132_Telicov_Letitia-Ioana-proiect%20(1).docx#_Toc135927145)

[CERINTA 4 12](file:///C:\Users\letit\OneDrive\Desktop\facultate\132_Telicov_Letitia-Ioana-proiect%20(1).docx#_Toc135927146)

[CERINTA 5 15](file:///C:\Users\letit\OneDrive\Desktop\facultate\132_Telicov_Letitia-Ioana-proiect%20(1).docx#_Toc135927147)

[CERINTA 6 22](file:///C:\Users\letit\OneDrive\Desktop\facultate\132_Telicov_Letitia-Ioana-proiect%20(1).docx#_Toc135927148)

[CERINTA 7 26](file:///C:\Users\letit\OneDrive\Desktop\facultate\132_Telicov_Letitia-Ioana-proiect%20(1).docx#_Toc135927149)

[CERINTA 8 28](file:///C:\Users\letit\OneDrive\Desktop\facultate\132_Telicov_Letitia-Ioana-proiect%20(1).docx#_Toc135927150)

[CERINTA 9 31](file:///C:\Users\letit\OneDrive\Desktop\facultate\132_Telicov_Letitia-Ioana-proiect%20(1).docx#_Toc135927151)

[CERINTA 10 33](file:///C:\Users\letit\OneDrive\Desktop\facultate\132_Telicov_Letitia-Ioana-proiect%20(1).docx#_Toc135927152)

[CERINTA 11 34](file:///C:\Users\letit\OneDrive\Desktop\facultate\132_Telicov_Letitia-Ioana-proiect%20(1).docx#_Toc135927153)

[CERINTA 12 35](file:///C:\Users\letit\OneDrive\Desktop\facultate\132_Telicov_Letitia-Ioana-proiect%20(1).docx#_Toc135927154)

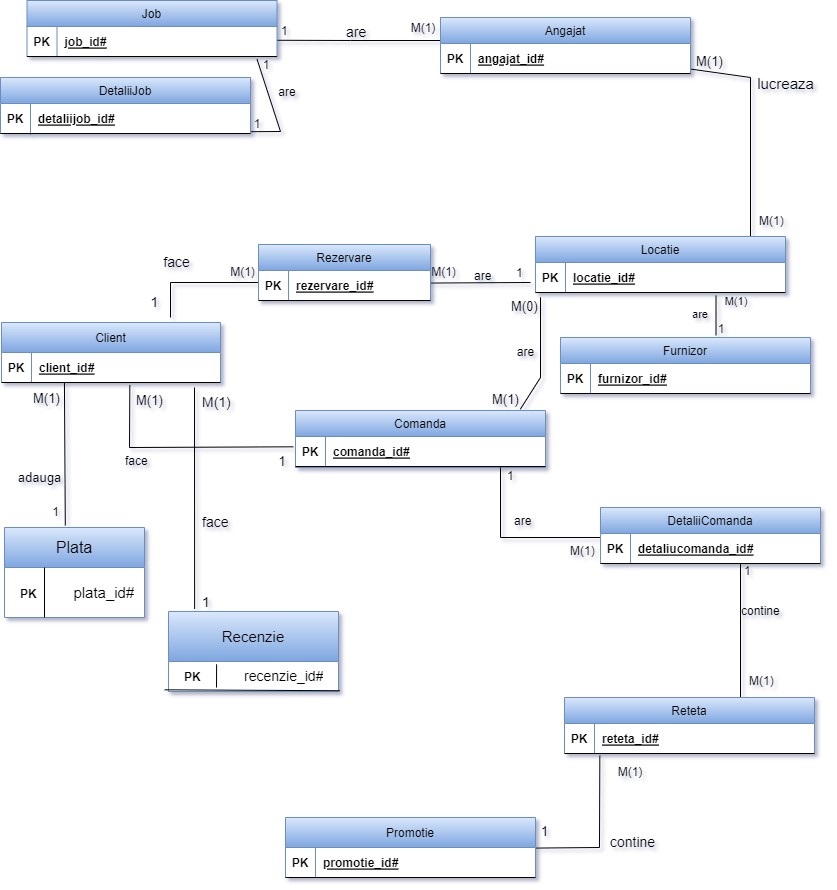
[CERINTA 13 36](file:///C:\Users\letit\OneDrive\Desktop\facultate\132_Telicov_Letitia-Ioana-proiect%20(1).docx#_Toc135927155)

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

Cofetăria are mai multe locații amplasate în diferite zone din București. La fiecare dintre aceste locații lucreaza mai mulți angajați, angajatul neaparținând unei locații în mod special. Fiecare angajat ocupă un post, dar pot fi mai mulți angajați care au aceeași funcție. Clienții cofetăriei trebuie sa facă o rezervare pentru a se putea bucura de serviciile acesteia. Fiecare poate da o comanda,care este constituită din mai multe detalii, cum ar fi numele, prețul acesteia și cantitatea dorită. De asemenea,multe din rețele făcute beneficiază și de promoții care sunt stabilite pentru o perioadă anume. Clienții pot lăsa si o recenzie pentru cofetărie și anume un rating de la 1 la 10.

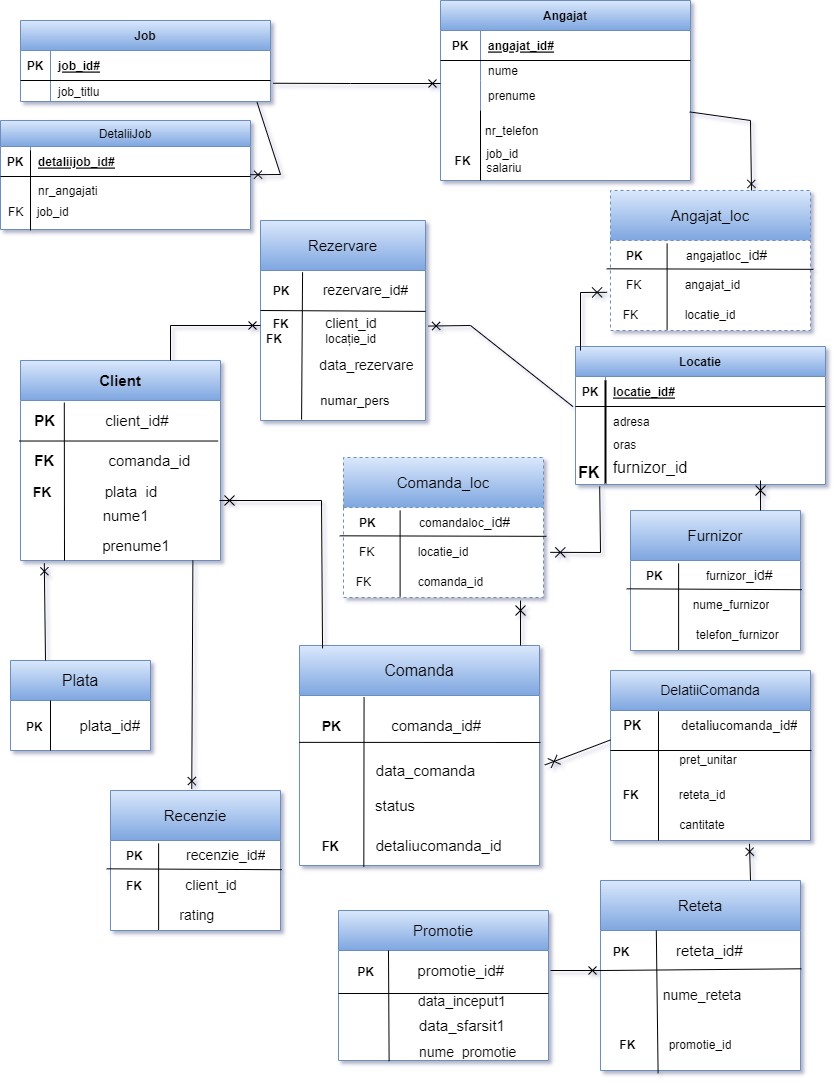
1. Realizați **diagrama entitate-relație** (ERD): entitățile, relațiile și atributele trebuie definite în limba română (vezi curs SGBD / model de diagrama ERD; nu se va accepta alt format).

DIAGRAMA ENTITATE-RELAȚIE



1. Pornind de la diagrama entitate-relație realizați **diagrama conceptuală** a modelului propus, integrând toate atributele necesare: entitățile, relațiile și atributele trebuie definite în limba română.

DIAGRAMA CONCEPTUALĂ



# Descriere entități

* ANGAJAT
* Persoană fizică care lucrează în cofetărie pe un anumit post
* Entitate dependentă de JOB
* Cheie primară: angajat\_id
* JOB
* Funcții pe care o are le au angajații cofetăriei
* Entitate independentă
* Cheie primară: job\_id
* DETALIIJOB
* Reprezintă informații despre fiecare job, cum ar fi câți angajați are

postul respectiv

* Entitate dependentă de JOB
* Cheie primară:detaliijob\_id
* CLIENT
* Persoană fizică care beneficiază de serviciile cofetăriei
* Entitate dependentă de COMANDA și PLATĂ
* Cheie primară:client\_id
* REZERVARE
* Acțiune realizată de un client
* Entitate dependentă de CLIENT și LOCAȚIE
* Cheie primară: rezervare\_id
* LOCAȚIE
* Reprezintă adresele unde se află cofetăria
* Entitate dependentă de FURNIZOR
* Cheie primară: locație\_id
* FURNIZOR
* Reprezintă numele și numărul de telefon al furnizorului
* Entitate independentă
* Cheie primară: furnizor\_id
* PLATĂ
* Reprezintă id-ul plății făcute de un client
* Entitate independentă
* Cheie primară: plata\_id
* RECENZIE
* Reprezintă ratingul făcut de fiecare client
* Entitate dependentă de CLIENT
* Cheie primară: recenzie\_id
* COMANDĂ
* Reprezintă comanda dată de un client,cuprinzând informații despre statul,dată la care a fost făcută
* Entitate dependentă de LOCAȚIE și de DETALIICOMANDA
* Cheie primară: comanda\_id
* DETALIICOMANDA
* Reprezintă statutul fiecărei comenzi,prețul acesteia și numele
* Entitate dependentă de REȚETĂ
* Cheie primară: detaliucomanda\_id
* REȚETĂ
* Reprezintă numele rețetelor care sunt făcute în cofetărie
* Entitate dependentă de PROMOȚIE
* Cheie primară: rețetă\_id
* PROMOȚIE
* Reprezintă diferite promoții ce se aplică la diferite rețete
* Entitate independentă
* Cheie primară: promotie\_id

# Descriere relații

ANGAJAT\_are\_un\_JOB = relație între angajat și job.

Un ANGAJAT poate avea doar un JOB.

Un JOB poate fi ales de unul sau mai mulți ANGAJAȚI.

-Cardinalitate minimă: 1:1.

-Cardinalitate maximă: M:1.

JOB\_are\_DETALIIJOB= relație între job și detalii.

Un JOB poate avea un detaliu.

DETALIIJOB poate avea câte un JOB.

-Cardinalitate minimă: 1:1.

-Cardinalitate maximă: 1:1.

ANGAJAT\_lucrează\_LOCAȚIE = relație între angajat și locație.

Un ANGAJAT poate lucra într-o LOCAȚIE sau în mai multe.

O LOCAȚIE poate avea mai unul sau mai mulți angajați.

-Cardinalitate minimă: 1:1.

-Cardinalitate maximă: M:M.

CLIENT\_face\_REZERVARE = relație între client și rezervare.

Un CLIENT poate face una sau mai multe rezervări.

O REZERVARE poate fi făcută de un CLIENT.

-Cardinalitate minimă: 1:1.

-Cardinalitate maximă: M:1.

LOCAȚIE\_are\_REZERVARE = relația între locație și rezervare.

O LOCAȚIE poate avea una sau mai multe rezervări.

O REZERVARE poate avea o locație.

-Cardinalitate minimă: 1:1.

-Cardinalitate maximă: M:1.

LOCAȚIE\_are\_FURNIZOR = relația între locație și furnizor.

O LOCAȚIE are un singur furnizor.

UN FURNIZOR poate avea o locație sau mai multe.

-Cardinalitate minimă: 1:1.

-Cardinalitate maximă: M:1.

CLIENT\_adauga\_PLATA = relație între client și plată.

Un CLIENT poate adăuga câte o singură PLATĂ.

O PLATĂ poate fi adăugată de un singur CLIENT sau mai mulți.

-Cardinalitate minimă: 1:1.

-Cardinalitate maximă: M:1.

CLIENT\_are\_RECENZIE = relație între client și recenzie.

Un CLIENT poate avea o singură RECENZIE.

O RECENZIE poate fi făcută de un singur client sau de mulți clienți.

-Cardinalitate minimă: 1:1.

-Cardinalitate maximă: M:1.

CLIENT\_adauga\_COMANDA = relația între client și comandă.

Un CLIENT poate adăuga câte o COMANDĂ.

O COMANDĂ poate fi adăugată de unul sau mai mulți clienți.

-Cardinalitate minimă: 1:1.

-Cardinalitate maximă: M:1.

COMANDA\_are\_DETALIICOMANDA = relație între comandă și detalii.

O COMANDĂ poate avea unul sau mai multe detalii.

DETALIICOMANDĂ poate avea câte o singură COMANDĂ.

-Cardinalitate minimă: 1:1.

-Cardinalitate maximă: M:1.

COMANDA\_aparține\_LOCAȚIE = relație între comandă și locație.

O COMANDĂ poate avea una sau mai multe locații.

O LOCAȚIE poate avea 0 comenzii sau mai multe.

-Cardinalitate minimă: 0:1.

-Cardinalitate maximă: M:M.

DETALIICOMANDA\_contine\_RETETA = relația dintre detaliile comenzii și rețetă.

DETALIICOMANDĂ poate conține una sau mai multe rețete.

O REȚETĂ poate fi adăugată doar o dată într-o comandă.

-Cardinalitate minimă: 1:1.

-Cardinalitate maximă: M:1.

PROMOTIE\_contine\_RETETA = relație între promoție și rețetă.

O PROMOȚIE poate conține una sau mai multe rețete.

O REȚETĂ poate aparține unei singure promoții.

-Cardinalitate minimă: 1:1.

-Cardinalitate maximă: M:1.

# Descriere atribute

Toate atributele au constrângerea NOT NULL.

* ANGAJAT

- angajat\_ID = cod numeric unic, de tip number, de 6 cifre pentru întregistrarea codului unui angajat.

- job\_ID = cheie străină din tabela Job.

- nume = variabilă de tip varchar2, de lungime 50 .

- prenume= variabilă de tip varchar2, de lungime 50 .

- nr\_telefon = variabilă de tip number, de lungime 10 .

- salariu= variabilă de tip number , de lungime 5 .

* JOB

- job\_ID = cod numeric unic, de tip number, de 6 cifre pentru întregistrarea codului unui job.

- job\_titlu = variabilă de tip varchar2, de lungime 50 .

* DETALIIJOB

- detaliijob\_ID = cod numeric unic, de tip number, de 6 cifre pentru întregistrarea codului unui detaliu.

- job\_ID = cheie străină din tabela Job.

-nr\_angajați= variabilă de tip number, de lungime 5.

* LOCAȚIE

- locatie\_ID = cod numeric unic, de tip number, de 6 cifre pentru întregistrarea codului unei locații.

- furnizor\_ID = cheie străină din tabela Furnizor.

-adresa = variabilă de tip varchar2, de lungime 50 .

-oras = variabilă de tip varchar2, de lungime 50 .

* FURNIZOR

- furnizor\_ID = cod numeric unic, de tip number, de 6 cifre pentru întregistrarea codului unui furnizor.

-nume\_furnizor = variabilă de tip varchar2, de lungime 50 .

-telefon\_furnizor = variabilă de tip varchar2, de lungime 50 .

* CLIENT

- client\_ID = cod numeric unic, de tip number, de 6 cifre pentru întregistrarea codului unei client.

- comanda\_ID = cheie străină din tabela Comanda.

- plata\_ID = cheie străină din tabela Plata.

-nume1= variabilă de tip varchar2, de lungime 50 .

-prenume1 = variabilă de tip varchar2, de lungime 50 .

* REZERVARE

- rezervare\_ID = cod numeric unic, de tip number, de 6 cifre pentru întregistrarea codului unei rezervări.

- client\_ID = cheie străină din tabela Client.

- locatie\_ID = cheie străină din tabela Locatie.

-numar\_pers= variabilă de tip number.

-data\_rezervare = variabilă de tip date.

* PLATA

- plata\_ID = cod numeric unic, de tip number, de 6 cifre pentru întregistrarea codului unei plăți.

* RECENZIE

- recenzie\_ID = cod numeric unic, de tip number, de 6 cifre pentru întregistrarea codului unei recenzii.

-rating = variabilă de tip number.

* COMANDA

- comanda\_ID = cod numeric unic, de tip number, de 6 cifre pentru întregistrarea codului unei comenzi.

- detaliucomanda\_ID = cheie străină din tabela DetaliiComanda.

-status= variabilă de tip varchar2, de lungime 50 .

-data\_comanda= variabilă de tip date.

* DETALIICOMANDA

- detaliucomanda\_ID = cod numeric unic, de tip number, de 6 cifre pentru întregistrarea codului unui detaliu.

- reteta\_ID = cheie străină din tabela Reteta.

-pret\_unitar= variabilă de tip number, de lungime 5.

-cantitate= variabilă de tip number, de lungime 5.

* RETETA

- reteta\_ID = cod numeric unic, de tip number, de 6 cifre pentru întregistrarea codului unei rețete.

-nume\_reteta= variabilă de tip varchar2, de lungime 50 .

- promotie\_ID = cheie străină din tabela Promotie.

* PROMOTIE

- promotie\_ID = cod numeric unic, de tip number, de 6 cifre pentru întregistrarea codului unei promotii.

-nume\_promotie= variabilă de tip varchar2, de lungime 50 .

-data\_inceput1 = variabilă de tip date.

-data\_sfarsit1 = variabilă de tip date.

* ANGAJAT\_LOC

- angajatloc\_ID = cod numeric unic, de tip number, de 6 cifre.

- angajat\_ID = cheie străină din tabela Angajat.

- locație\_ID = cheie străină din tabela Locatie.

* COMANDA\_LOC

- comandaloc\_ID = cod numeric unic, de tip number, de 6 cifre.

- comanda\_ID = cheie străină din tabela Comanda.

- locatie\_ID = cheie străină din tabela Locatie.

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

create table Job (

    job\_id number(6) constraint pk\_job primary key,

    job\_titlu varchar2(50) constraint job\_titlu\_null not null

);

create table DetaliiJob(

    detaliijob\_id number(6) constraint pk\_detaliijob primary key,

    nr\_angajati number(5) constraint nr\_angajati\_null not null ,

    job\_id number(6) not null,

    constraint fk\_job foreign key (job\_id) references Job(job\_id)

);

create table Angajat(

    angajat\_id number(6) constraint pk\_angajat primary key,

    nume varchar2(50) constraint nume\_null not null ,

    prenume varchar2(50) constraint prenume\_null not null ,

    nr\_telefon number(16) constraint nr\_telefon\_null not null ,

    job\_id number(6) not null,

    salariu number(5) constraint salariu\_null not null ,

    constraint fk\_job\_v foreign key (job\_id) references Job(job\_id)

);

create table Locatie(

    locatie\_id number(6) constraint pk\_locatie primary key,

    adresa varchar2(50) constraint adresa\_null not null ,

    oras  varchar2(50) constraint oras\_null not null ,

    furnizor\_id number(6) not null,

    constraint fk\_furnizor foreign key (furnizor\_id) references Furnizor(furnizor\_id)

);

create table Rezervare(

    rezervare\_id number(6) constraint pk\_rezervare primary key,

    client\_id number(6) not null,

    locatie\_id number(6) not null,

    data\_rezervare date,

    numar\_pers number(30) constraint numar\_pers\_null not null ,

    constraint fk\_client\_r foreign key (client\_id) references Client(client\_id),

    constraint fk\_locatie\_r foreign key (locatie\_id) references Locatie(locatie\_id)

);

create table Client(

     client\_id number(6) constraint pk\_client primary key,

     comanda\_id number(6) not null,

     plata\_id number(6) not null,

     nume1 varchar2(50) constraint nume1\_null not null ,

     prenume1 varchar2(50) constraint prenume1\_null not null ,

     constraint fk\_comanda foreign key (comanda\_id) references Comanda(comanda\_id),

     constraint fk\_plata foreign key (plata\_id) references Plata(plata\_id)

);

create table Comanda(

     comanda\_id number(6) constraint pk\_comanda primary key,

     data\_comanda date,

     status varchar2(50) constraint status\_null not null ,

     detaliucomanda\_id number(6) not null,

     constraint fk\_detaliucomanda foreign key (detaliucomanda\_id) references DetaliiComanda(detaliucomanda\_id)

);

create table DetaliiComanda(

    detaliucomanda\_id number(6) constraint pk\_detaliucomanda primary key,

    pret\_unitar number(5) constraint pret\_unitar\_null not null ,

    reteta\_id number(6) not null,

    cantitate number(5) constraint cantitate\_null not null ,

    constraint fk\_reteta foreign key (reteta\_id) references Reteta(reteta\_id)

);

create table Plata(

    plata\_id number(6) constraint pk\_plata primary key

);

create table Recenzie(

    recenzie\_id number(6) constraint pk\_recenzie primary key,

    client\_id number(6) not null,

    rating number(16) constraint rating\_null not null,

    constraint fk\_client\_v foreign key (client\_id) references Client(client\_id)

);

create table Promotie(

    promotie\_id number(6) constraint pk\_promotie primary key,

    data\_inceput1 date  ,

    data\_sfarsit1 date not null ,

nume\_promotie varchar2(50) constraint nume\_promotie\_null not null

);

create table Reteta(

    reteta\_id number(6) constraint pk\_reteta primary key,

    nume\_reteta varchar2(50) constraint nume\_reteta\_null not null,

    promotie\_id number(6) not null,

    constraint fk\_promotie foreign key (promotie\_id) references Promotie(promotie\_id)

);

create table Angajat\_loc(

    angajatloc\_id number(6) constraint pk\_angajatloc primary key,

    angajat\_id number(6) not NULL,

    locatie\_id number(6) not NULL,

    constraint fk\_angajat\_a foreign key (angajat\_id) references Angajat(angajat\_id),

    constraint fk\_locatie\_a foreign key (locatie\_id) references Locatie(locatie\_id)

);

create table Comanda\_loc(

    comandaloc\_id number(6) constraint pk\_comandaloc primary key,

    locatie\_id number(6) not NULL,

    comanda\_id number(6) not NULL,

    constraint fk\_locatie\_c foreign key (locatie\_id) references Locatie(locatie\_id),

    constraint fk\_comanda\_c foreign key (comanda\_id) references Comanda(comanda\_id)

);

create table Furnizor(

    furnizor\_id number(6) constraint pk\_furnizor primary key,

    telefon\_furnizor varchar2(50) constraint telefon\_furnizor\_null not null ,

    nume\_furnizor varchar2(50) constraint nume\_furnizor\_null not null

);

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

insert into Angajat(angajat\_id, nume, prenume,nr\_telefon,job\_id,salariu) values (1001,'Cristescu','Lili',0768567892,10,2300);

insert into Angajat(angajat\_id, nume, prenume,nr\_telefon,job\_id,salariu) values (1002,'Georgescu','Marian',0756567892,11,1800);

insert into Angajat(angajat\_id, nume, prenume,nr\_telefon,job\_id,salariu) values (1003,'Patriche','Stefan',0756567522,12,1900);

insert into Angajat(angajat\_id, nume, prenume,nr\_telefon,job\_id,salariu) values (1004,'Soritu','Magda',0751267892,13,2000);

insert into Angajat(angajat\_id, nume, prenume,nr\_telefon,job\_id,salariu) values (1005,'Macrinu','Emilia',0756561092,14,2100);

insert into Angajat(angajat\_id, nume, prenume,nr\_telefon,job\_id,salariu) values (1006,'Bivoiu','Catalin',0786567892,10,2300);

insert into Angajat(angajat\_id, nume, prenume,nr\_telefon,job\_id,salariu) values (1007,'Alexandru','Anastasia',0756907892,14,2100);

insert into Angajat(angajat\_id, nume, prenume,nr\_telefon,job\_id,salariu) values (1008,'Ionisor','Sebastian',0756567412,15,2500);

insert into Angajat(angajat\_id, nume, prenume,nr\_telefon,job\_id,salariu) values (1009,'Iliasu','Corina',0797567892,12,1900);

insert into Angajat(angajat\_id, nume, prenume,nr\_telefon,job\_id,salariu) values (1010,'Mehedinti','George',0756564492,13,2000);

insert into Job(job\_id,job\_titlu) values (10,'Cofetar');

insert into Job(job\_id,job\_titlu) values (11,'Casier');

insert into Job(job\_id,job\_titlu) values (12,'Personal servire');

insert into Job(job\_id,job\_titlu) values (13,'Barista');

insert into Job(job\_id,job\_titlu) values (14,'Decorator torturi');

insert into Job(job\_id,job\_titlu) values (15,'Manager');

insert into DetaliiJob(detaliijob\_id,nr\_angajati,job\_id) values (100,2,10);

insert into DetaliiJob(detaliijob\_id,nr\_angajati,job\_id) values (101,1,11);

insert into DetaliiJob(detaliijob\_id,nr\_angajati,job\_id) values (102,2,12);

insert into DetaliiJob(detaliijob\_id,nr\_angajati,job\_id) values (103,2,13);

insert into DetaliiJob(detaliijob\_id,nr\_angajati,job\_id) values (104,2,14);

insert into DetaliiJob(detaliijob\_id,nr\_angajati,job\_id) values (105,1,15);

insert into Locatie(locatie\_id,adresa,oras,furnizor\_id) values (1,'Strada Bisnit nr. 8','Bucuresti',700);

insert into Locatie(locatie\_id,adresa,oras,furnizor\_id) values (2,'Strada Colos nr. 26','Bucuresti',701);

insert into Locatie(locatie\_id,adresa,oras,furnizor\_id) values (3,'Strada Alescu nr. 12','Bucuresti',702);

insert into Locatie(locatie\_id,adresa,oras,furnizor\_id) values (4,'Strada Democratiei nr. 7','Bucuresti',703);

insert into Locatie(locatie\_id,adresa,oras,furnizor\_id) values (5,'Strada Razboieni nr. 13','Bucuresti',704);

insert into Furnizor(furnizor\_id,nume\_furnizor,telefon\_furnizor) values (700,'CrissFoods',0745520552);

insert into Furnizor(furnizor\_id,nume\_furnizor,telefon\_furnizor) values (701,'Boromir',0745520523);

insert into Furnizor(furnizor\_id,nume\_furnizor,telefon\_furnizor) values (702,'Pakmaia',0745520232);

insert into Furnizor(furnizor\_id,nume\_furnizor,telefon\_furnizor) values (703,'Agrana',0790520552);

insert into Furnizor(furnizor\_id,nume\_furnizor,telefon\_furnizor) values (704,'Raureni',0745544552);

insert into Client(client\_id,comanda\_id,plata\_id,nume1,prenume1) values (60,600,90,'Constantin','Eusebiu');

insert into Client(client\_id,comanda\_id,plata\_id,nume1,prenume1) values (61,601,91,'Marinache','Cristina');

insert into Client(client\_id,comanda\_id,plata\_id,nume1,prenume1) values (62,602,92,'Mocanu','Claudiu');

insert into Client(client\_id,comanda\_id,plata\_id,nume1,prenume1) values (63,603,93,'Caragea','Mihaela');

insert into Client(client\_id,comanda\_id,plata\_id,nume1,prenume1) values (64,604,94,'Miron','Robert');

insert into Rezervare(rezervare\_id,client\_id,locatie\_id,data\_rezervare,numar\_pers) values (70,60,1,'12-JUN-2023',10);

insert into Rezervare(rezervare\_id,client\_id,locatie\_id,data\_rezervare,numar\_pers) values (71,61,2,'10-JUL-2023',11);

insert into Rezervare(rezervare\_id,client\_id,locatie\_id,data\_rezervare,numar\_pers) values (72,62,3,'23-JUN-2023',12);

insert into Rezervare(rezervare\_id,client\_id,locatie\_id,data\_rezervare,numar\_pers) values (73,63,4,'11-JUL-2023',21);

insert into Rezervare(rezervare\_id,client\_id,locatie\_id,data\_rezervare,numar\_pers) values (74,64,5,'12-JUL-2023',7);

insert into Comanda(comanda\_id,data\_comanda,status,detaliucomanda\_id) values (600,'28-JUN-2023','in proces',30);

insert into Comanda(comanda\_id,data\_comanda,status,detaliucomanda\_id) values (601,'30-DEC-2023','finalizata',31);

insert into Comanda(comanda\_id,data\_comanda,status,detaliucomanda\_id) values (602,'09-JUL-2023','in proces',32);

insert into Comanda(comanda\_id,data\_comanda,status,detaliucomanda\_id) values (603,'29-JUN-2023','finalizata',33);

insert into Comanda(comanda\_id,data\_comanda,status,detaliucomanda\_id) values (604,'29-JUN-2023','in proces',34);

insert into DetaliiComanda(detaliucomanda\_id,pret\_unitar,reteta\_id,cantitate) values (30,12,40,1);

insert into DetaliiComanda(detaliucomanda\_id,pret\_unitar,reteta\_id,cantitate) values (31,54,41,3);

insert into DetaliiComanda(detaliucomanda\_id,pret\_unitar,reteta\_id,cantitate) values (32,70,42,1);

insert into DetaliiComanda(detaliucomanda\_id,pret\_unitar,reteta\_id,cantitate) values (33,40,43,2);

insert into DetaliiComanda(detaliucomanda\_id,pret\_unitar,reteta\_id,cantitate) values (34,35,44,4);

insert into DetaliiComanda(detaliucomanda\_id,pret\_unitar,reteta\_id,cantitate) values (35,60,45,1);

insert into DetaliiComanda(detaliucomanda\_id,pret\_unitar,reteta\_id,cantitate) values (36,70,46,2);

insert into DetaliiComanda(detaliucomanda\_id,pret\_unitar,reteta\_id,cantitate) values (37,23,47,3);

insert into DetaliiComanda(detaliucomanda\_id,pret\_unitar,reteta\_id,cantitate) values (38,30,48,1);

insert into DetaliiComanda(detaliucomanda\_id,pret\_unitar,reteta\_id,cantitate) values (39,25,49,3);

insert into Plata(plata\_id) values (90);

insert into Plata(plata\_id) values (91);

insert into Plata(plata\_id) values (92);

insert into Plata(plata\_id) values (93);

insert into Plata(plata\_id) values (94);

insert into Recenzie(recenzie\_id,client\_id,rating) values (80,60,10);

insert into Recenzie(recenzie\_id,client\_id,rating) values (81,61,8);

insert into Recenzie(recenzie\_id,client\_id,rating) values (82,62,9);

insert into Recenzie(recenzie\_id,client\_id,rating) values (83,63,10);

insert into Recenzie(recenzie\_id,client\_id,rating) values (84,64,7);

insert into Promotie(promotie\_id,data\_inceput1,data\_sfarsit1,nume\_promotie) values (2000,'01-JUN-2023', '10-JUL-2023','cumperi un tort si primesti si un ornament gratis');

insert into Promotie(promotie\_id,data\_inceput1,data\_sfarsit1,nume\_promotie) values (2001,'16-JUN-2023', '25-JUN-2023','promotie 10% la orice produs');

insert into Promotie(promotie\_id,data\_inceput1,data\_sfarsit1,nume\_promotie) values (2002,'01-FEB-2023', '10-FEB-2023','promotie 5% la orice produs mic-dejun ');

insert into Promotie(promotie\_id,data\_inceput1,data\_sfarsit1,nume\_promotie) values (2003,'23-OCT-2023', '31-OCT-2023','cumperi un tort si al doilea ales este gratis');

insert into Promotie(promotie\_id,data\_inceput1,data\_sfarsit1,nume\_promotie) values (2004,'07-JUN-2023', '10-JUN-2023','cumpara un croissant/briosa si primesti o surpriza');

insert into Reteta(reteta\_id,nume\_reteta,promotie\_id) values (40,'Tort Red Velvet',2000);

insert into Reteta(reteta\_id,nume\_reteta,promotie\_id) values (41,'Tiramisu',2001);

insert into Reteta(reteta\_id,nume\_reteta,promotie\_id) values (42,'Macarons',2001);

insert into Reteta(reteta\_id,nume\_reteta,promotie\_id) values (43,'Briose',2004);

insert into Reteta(reteta\_id,nume\_reteta,promotie\_id) values (44,'Cheesecake cu fructe de pădure',2003);

insert into Reteta(reteta\_id,nume\_reteta,promotie\_id) values (45,'Eclair',2001);

insert into Reteta(reteta\_id,nume\_reteta,promotie\_id) values (46,'Tort Oreo',2000);

insert into Reteta(reteta\_id,nume\_reteta,promotie\_id) values (47,'Almond Joy',2001);

insert into Reteta(reteta\_id,nume\_reteta,promotie\_id) values (48,'Prajituri cu crema',2002);

insert into Reteta(reteta\_id,nume\_reteta,promotie\_id) values (49,'Croissant',2004);

insert into Angajat\_loc(angajatloc\_id,angajat\_id,locatie\_id) values(3000,1001,1);

insert into Angajat\_loc(angajatloc\_id,angajat\_id,locatie\_id) values(3001,1002,2);

insert into Angajat\_loc(angajatloc\_id,angajat\_id,locatie\_id) values(3002,1003,2);

insert into Angajat\_loc(angajatloc\_id,angajat\_id,locatie\_id) values(3003,1004,3);

insert into Angajat\_loc(angajatloc\_id,angajat\_id,locatie\_id) values(3004,1005,1);

insert into Angajat\_loc(angajatloc\_id,angajat\_id,locatie\_id) values(3005,1006,2);

insert into Angajat\_loc(angajatloc\_id,angajat\_id,locatie\_id) values(3006,1007,4);

insert into Angajat\_loc(angajatloc\_id,angajat\_id,locatie\_id) values(3007,1008,3);

insert into Angajat\_loc(angajatloc\_id,angajat\_id,locatie\_id) values(3008,1009,4);

insert into Angajat\_loc(angajatloc\_id,angajat\_id,locatie\_id) values(3009,1010,3);

insert into Comanda\_loc(comandaloc\_id,locatie\_id,comanda\_id) values (7000,2,600);

insert into Comanda\_loc(comandaloc\_id,locatie\_id,comanda\_id) values (7001,5,601);

insert into Comanda\_loc(comandaloc\_id,locatie\_id,comanda\_id) values (7002,4,601);

insert into Comanda\_loc(comandaloc\_id,locatie\_id,comanda\_id) values (7003,3,602);

insert into Comanda\_loc(comandaloc\_id,locatie\_id,comanda\_id) values (7004,4,600);

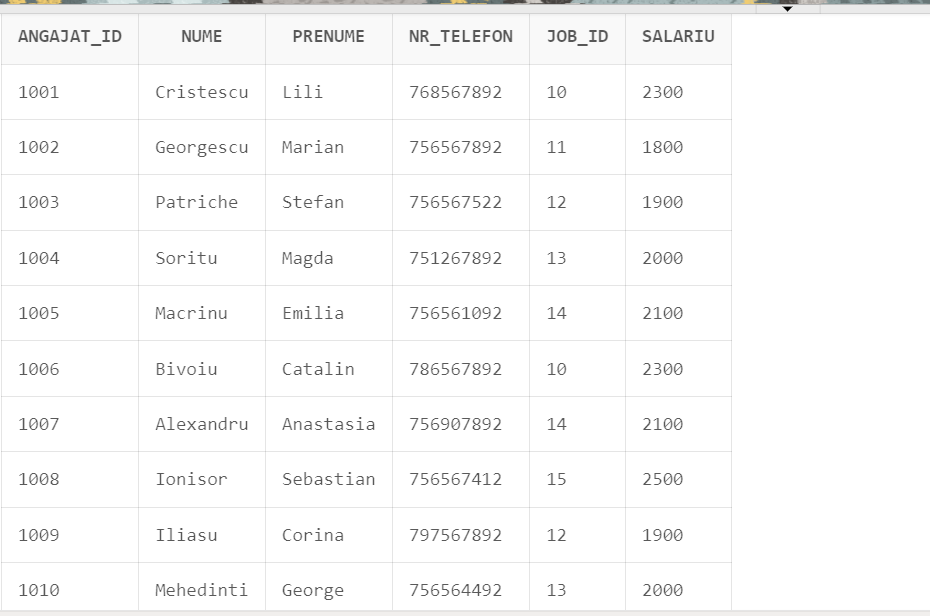
insert into Comanda\_loc(comandaloc\_id,locatie\_id,comanda\_id) values (7005,1,603);

insert into Comanda\_loc(comandaloc\_id,locatie\_id,comanda\_id) values (7006,5,603);

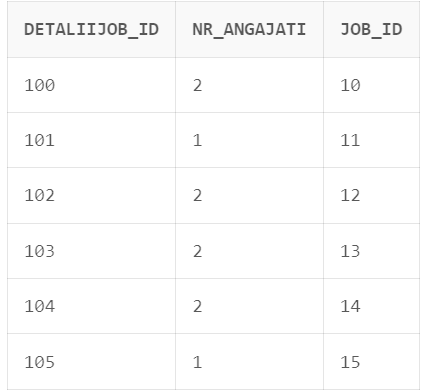
insert into Comanda\_loc(comandaloc\_id,locatie\_id,comanda\_id) values (7007,2,604);

insert into Comanda\_loc(comandaloc\_id,locatie\_id,comanda\_id) values (7008,1,604);

insert into Comanda\_loc(comandaloc\_id,locatie\_id,comanda\_id) values (7009,3,602);

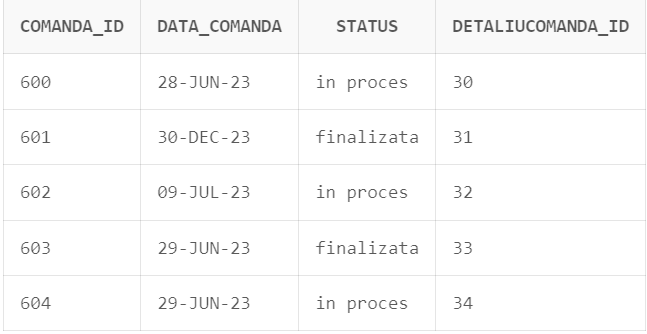


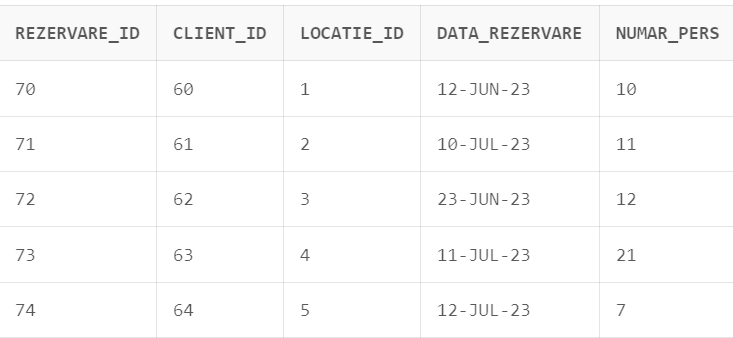
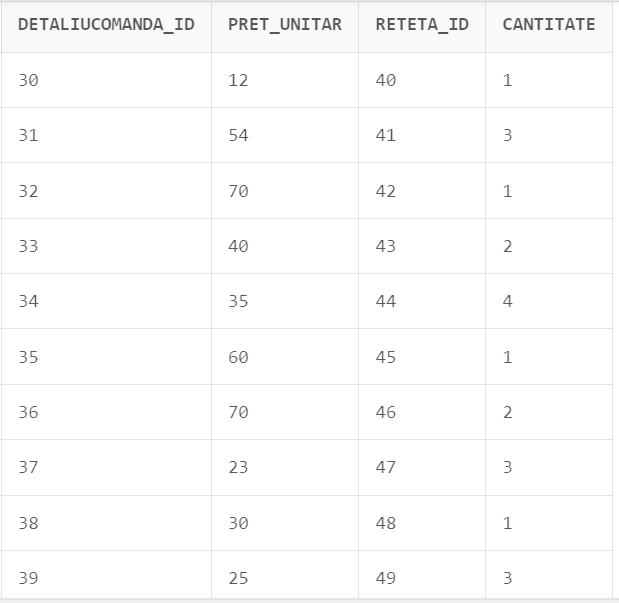
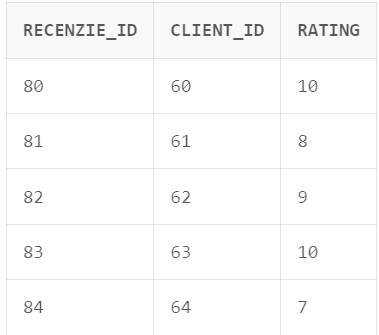




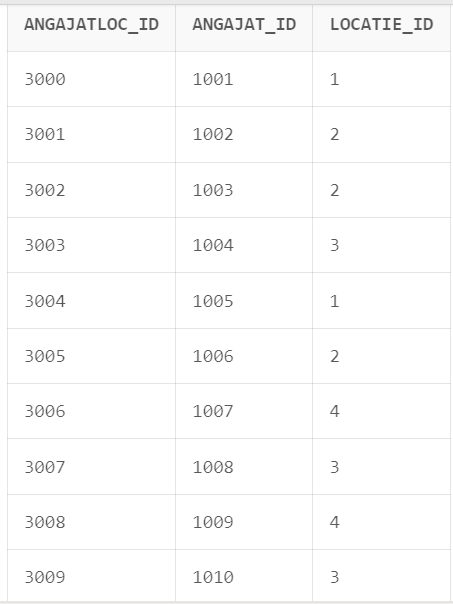


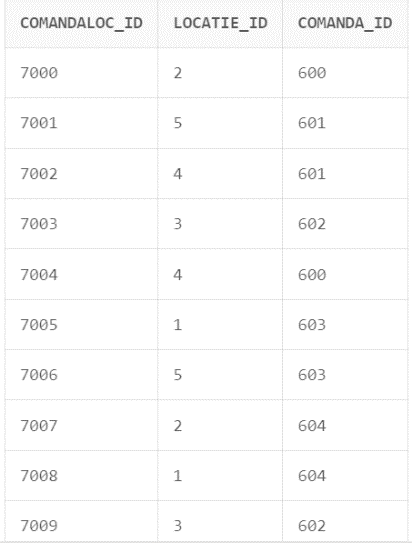
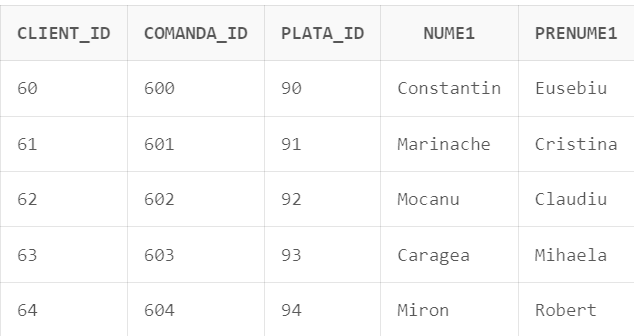












MODIFICAT

1. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un **subprogram stocat independent** care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.

Pentru fiecare rezervare făcută într-o anumită locație să se afișeze data la care au fost făcute, numărul angajaților care lucreaza în locația respectivă și numele clientului care a făcut rezervarea respectivă.

CREATE OR REPLACE PROCEDURE DisplayReservationDetails(

location\_id\_param IN NUMBER

)

IS

TYPE ReservationInfoType IS RECORD (

reservation\_date DATE,

employee\_count NUMBER,

client\_name VARCHAR2(50)

);

TYPE DateReservations IS TABLE OF DATE INDEX BY PLS\_INTEGER;

TYPE EmployeeCounts IS TABLE OF ReservationInfoType ;

TYPE ClientNames IS VARRAY(50) OF ReservationInfoType;

reservation\_dates DateReservations;

reservation\_employee\_counts EmployeeCounts;

reservation\_client\_names ClientNames;

BEGIN

reservation\_dates := DateReservations();

reservation\_employee\_counts := EmployeeCounts();

reservation\_client\_names := ClientNames();

FOR rec IN (

SELECT r.data\_rezervare, COUNT(DISTINCT al.angajat\_id) AS employee\_count, c.nume1 AS client\_name

FROM Rezervare r

JOIN Angajat\_Loc al ON r.locatie\_id = al.locatie\_id

JOIN Angajat a ON al.angajat\_id = a.angajat\_id

JOIN Client c ON r.client\_id = c.client\_id

WHERE r.locatie\_id = location\_id\_param

GROUP BY r.data\_rezervare, c.nume1

)

LOOP

reservation\_dates(reservation\_dates.COUNT + 1) := rec.data\_rezervare;

IF rec.employee\_count IS NOT NULL THEN

reservation\_employee\_counts.extend;

reservation\_employee\_counts(reservation\_employee\_counts.LAST). employee\_count := rec. employee\_count;

ELSE

reservation\_employee\_counts.extend;

reservation\_employee\_counts(reservation\_employee\_counts.LAST). employee\_count := 0;

END IF;

reservation\_client\_names.extend;

reservation\_client\_names(reservation\_client\_names.LAST).client\_name := rec.client\_name;

END LOOP;

FOR i IN reservation\_dates.FIRST..reservation\_dates.LAST

LOOP

DBMS\_OUTPUT.PUT\_LINE('Rezervare la data: ' || TO\_CHAR(reservation\_dates(i), 'DD-MON-YYYY'));

DBMS\_OUTPUT.PUT\_LINE('Numarul angajatilor care lucreaza în locatia respectiva: ' || reservation\_employee\_counts(i).employee\_count);

DBMS\_OUTPUT.PUT\_LINE('Numele clientului: ' || reservation\_client\_names(i).client\_name);

END LOOP;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu au fost gasite rezervari pentru locatia specificata.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare neasteptata: ' || SQLERRM);

END DisplayReservationDetails;

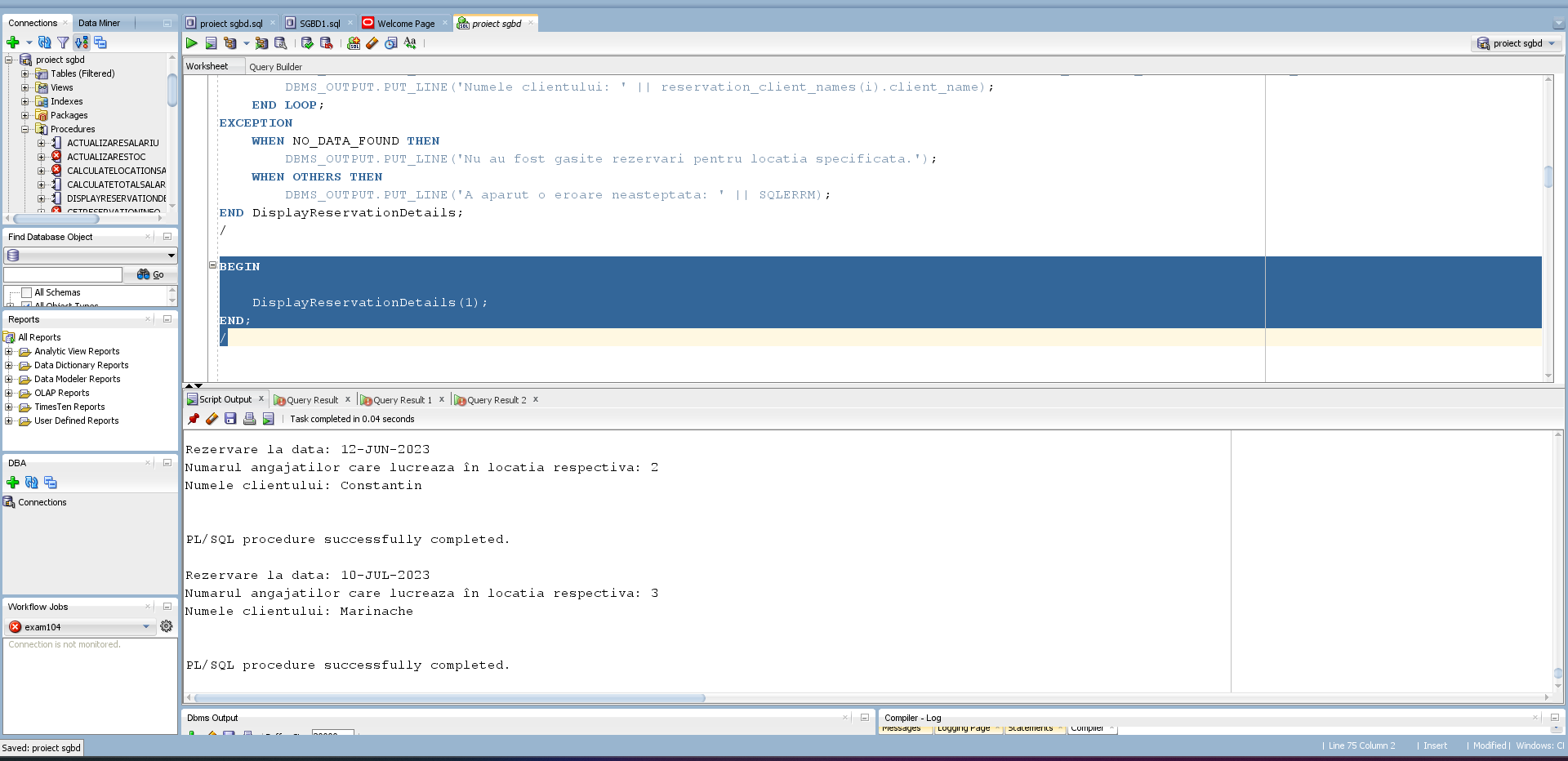
/

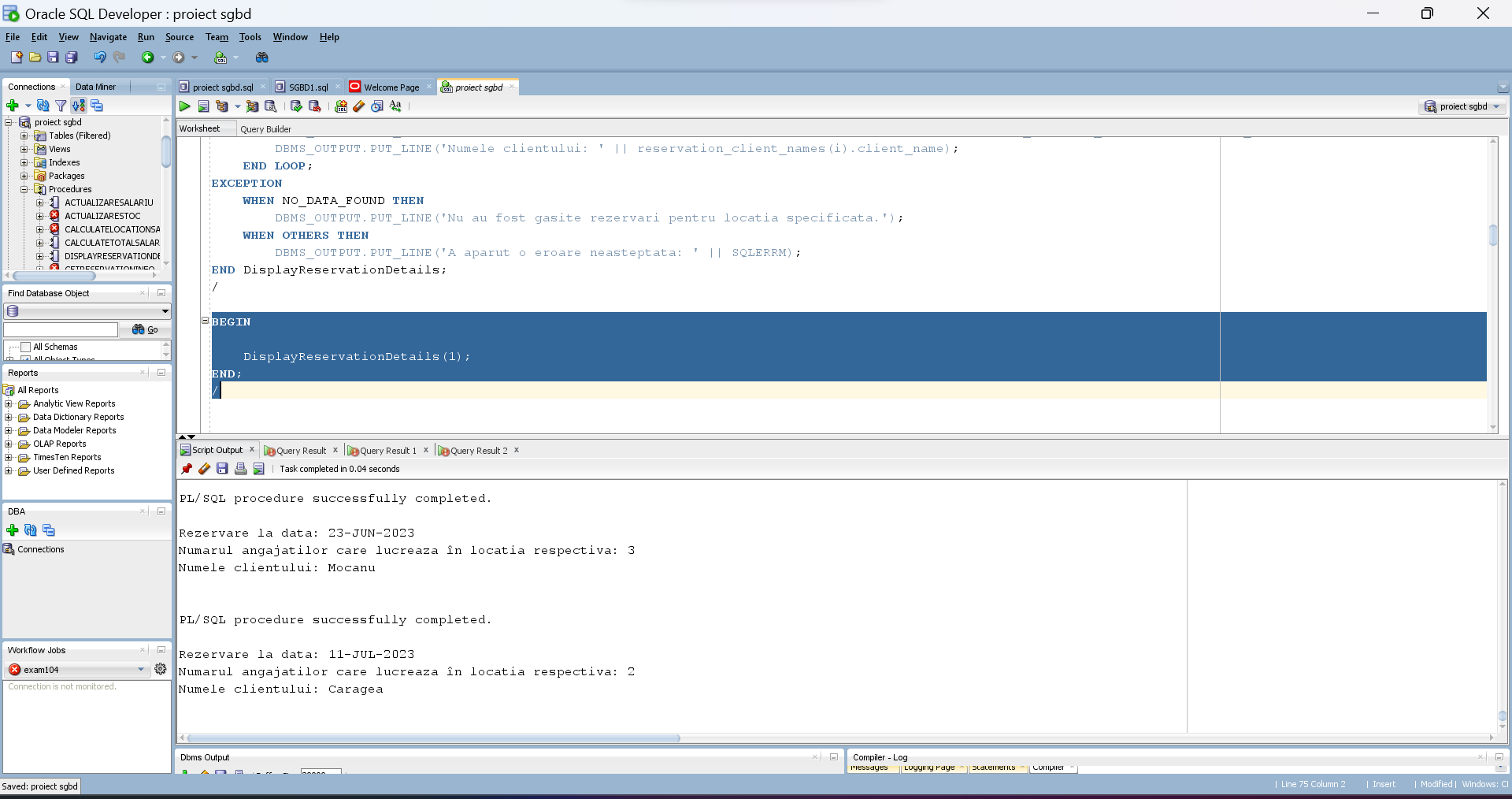
BEGIN

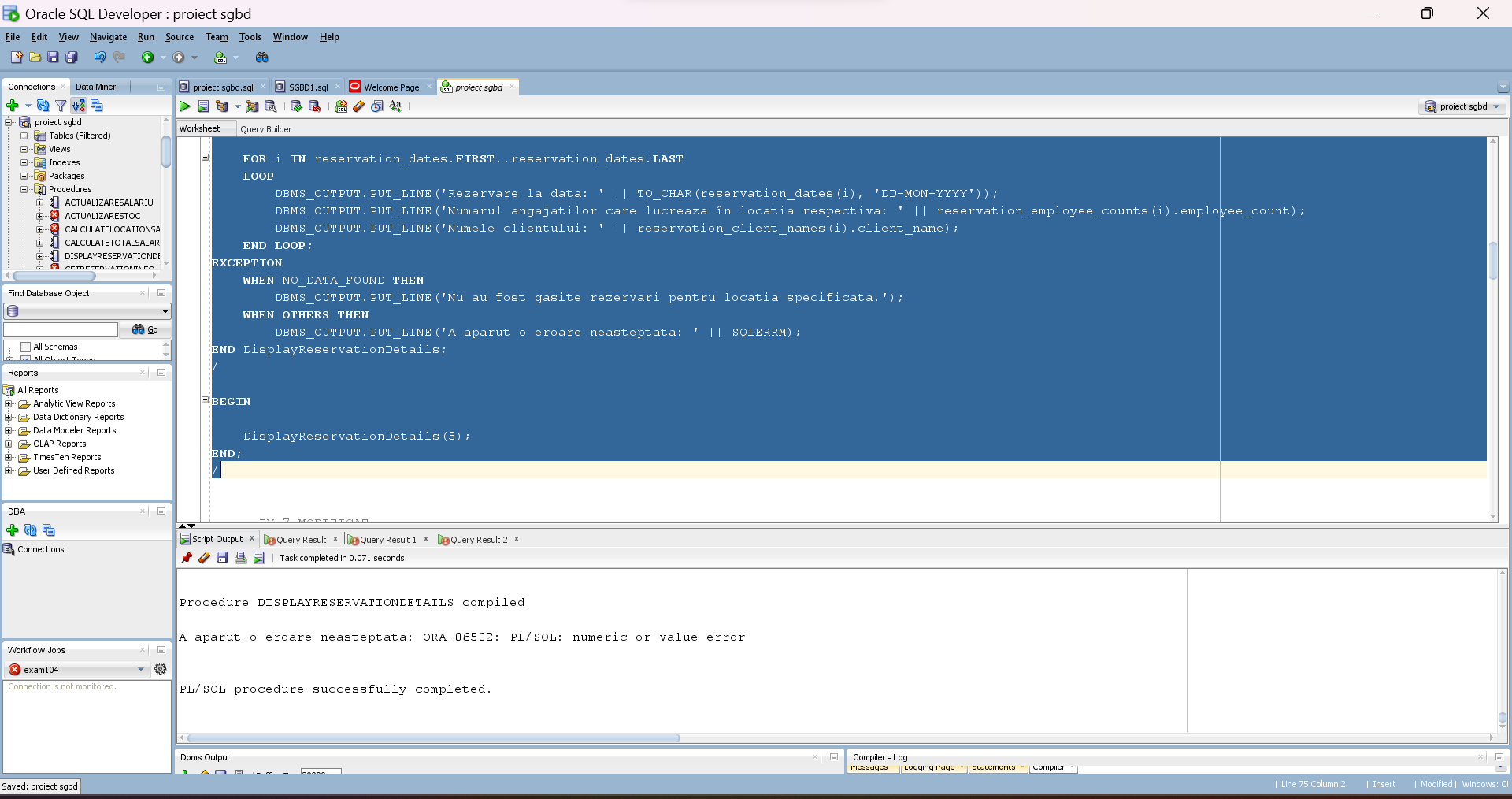
DisplayReservationDetails(1);

END;

/







MODIFICAT

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

Calculați suma salariilor angajaților pentru fiecare locație .

CREATE OR REPLACE PROCEDURE CalculateTotalSalariesForEachLocation AS

--CURSOR LocationCursor IS

-- SELECT DISTINCT locatie\_id

-- FROM Angajat\_Loc;

CURSOR SalaryCursor (location\_id NUMBER) IS

SELECT salariu

FROM Angajat\_Loc al

JOIN Angajat a ON al.angajat\_id = a.angajat\_id

WHERE al.locatie\_id = location\_id;

total\_salary NUMBER;

BEGIN

FOR loc IN

(SELECT DISTINCT locatie\_id

FROM Angajat\_Loc

)LOOP

total\_salary := 0;

FOR sal IN SalaryCursor(loc.locatie\_id) LOOP

total\_salary := total\_salary + sal.salariu;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Suma salariilor pentru locatia ' || loc.locatie\_id || ': ' || total\_salary);

END LOOP;

END CalculateTotalSalariesForEachLocation;

/

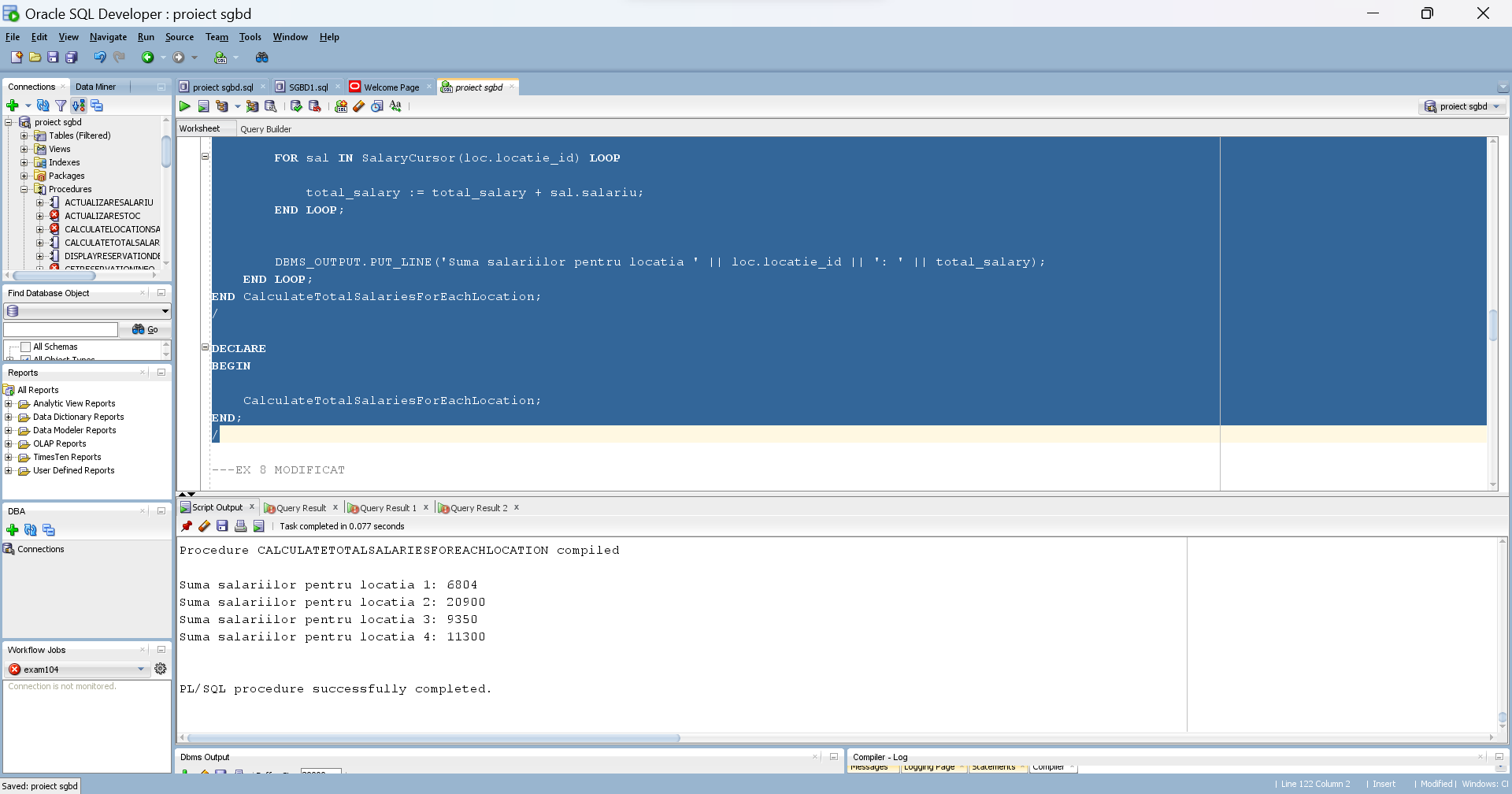
DECLARE

BEGIN

CalculateTotalSalariesForEachLocation;

END;

/



MODIFICAT

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

Calculați suma totală a comenzilor plasate la o anumită locație într-o perioadă specificată.

CREATE OR REPLACE FUNCTION CalculSumaComenziLocatie(

p\_locatie\_id IN NUMBER,

p\_data\_inceput IN DATE,

p\_data\_sfarsit IN DATE

) RETURN NUMBER IS

v\_suma NUMBER := 0;

v\_numar\_comenzi NUMBER := 0;

v\_row\_count NUMBER := 0;

-- Exceptii proprii

LocatieNotFound EXCEPTION;

InvalidDateRange EXCEPTION;

BEGIN

IF p\_data\_inceput > p\_data\_sfarsit THEN

RAISE InvalidDateRange;

END IF;

SELECT COUNT(\*)

INTO v\_numar\_comenzi

FROM Locatie l

JOIN Comanda\_Loc cl on l.locatie\_id = cl.locatie\_id

JOIN Comanda c ON cl.comanda\_id = c.comanda\_id

WHERE l.locatie\_id = p\_locatie\_id

AND c.data\_comanda BETWEEN p\_data\_inceput AND p\_data\_sfarsit;

IF v\_numar\_comenzi = 0 THEN

RAISE LocatieNotFound;

END IF;

SELECT SUM(dc.pret\_unitar \* dc.cantitate)

INTO v\_suma

FROM Comanda\_Loc cl

JOIN Comanda c ON cl.comanda\_id = c.comanda\_id

JOIN DetaliiComanda dc ON c.detaliucomanda\_id = dc.detaliucomanda\_id

WHERE cl.locatie\_id = p\_locatie\_id

AND c.data\_comanda BETWEEN p\_data\_inceput AND p\_data\_sfarsit;

RETURN v\_suma;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista date.');

WHEN InvalidDateRange THEN

DBMS\_OUTPUT.PUT\_LINE('Intervalul de date este invalid.');

RETURN NULL;

WHEN LocatieNotFound THEN

DBMS\_OUTPUT.PUT\_LINE('Locatia nu are comenzi preluate in perioada respectiva.');

RETURN NULL;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('O eroare neasteptata a aparut.');

RETURN NULL;

END CalculSumaComenziLocatie;

/

DECLARE

v\_locatie\_id NUMBER := 5;

v\_data\_inceput DATE := TO\_DATE('10-JAN-2024', 'DD-MON-YYYY');

v\_data\_sfarsit DATE := TO\_DATE('30-DEC-2024', 'DD-MON-YYYY');

v\_suma\_comenzi NUMBER;

BEGIN

v\_suma\_comenzi := CalculSumaComenziLocatie(v\_locatie\_id, v\_data\_inceput, v\_data\_sfarsit);

IF v\_suma\_comenzi IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Suma comenzilor pentru locatia ' || v\_locatie\_id || ' în perioada ' ||

TO\_CHAR(v\_data\_inceput, 'DD-MON-YYYY') || ' - ' ||

TO\_CHAR(v\_data\_sfarsit, 'DD-MON-YYYY') || ' este: ' || v\_suma\_comenzi);

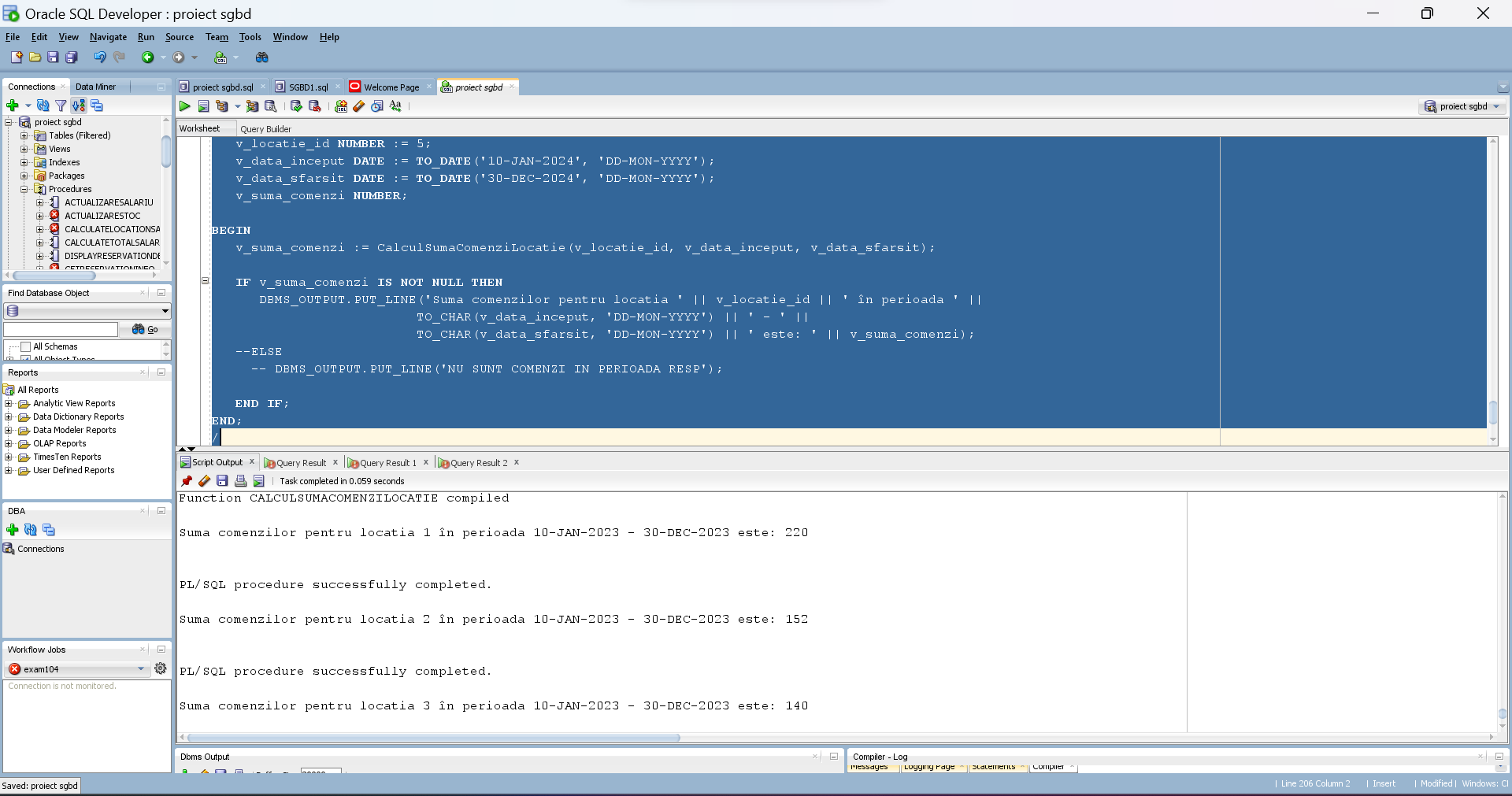
--ELSE

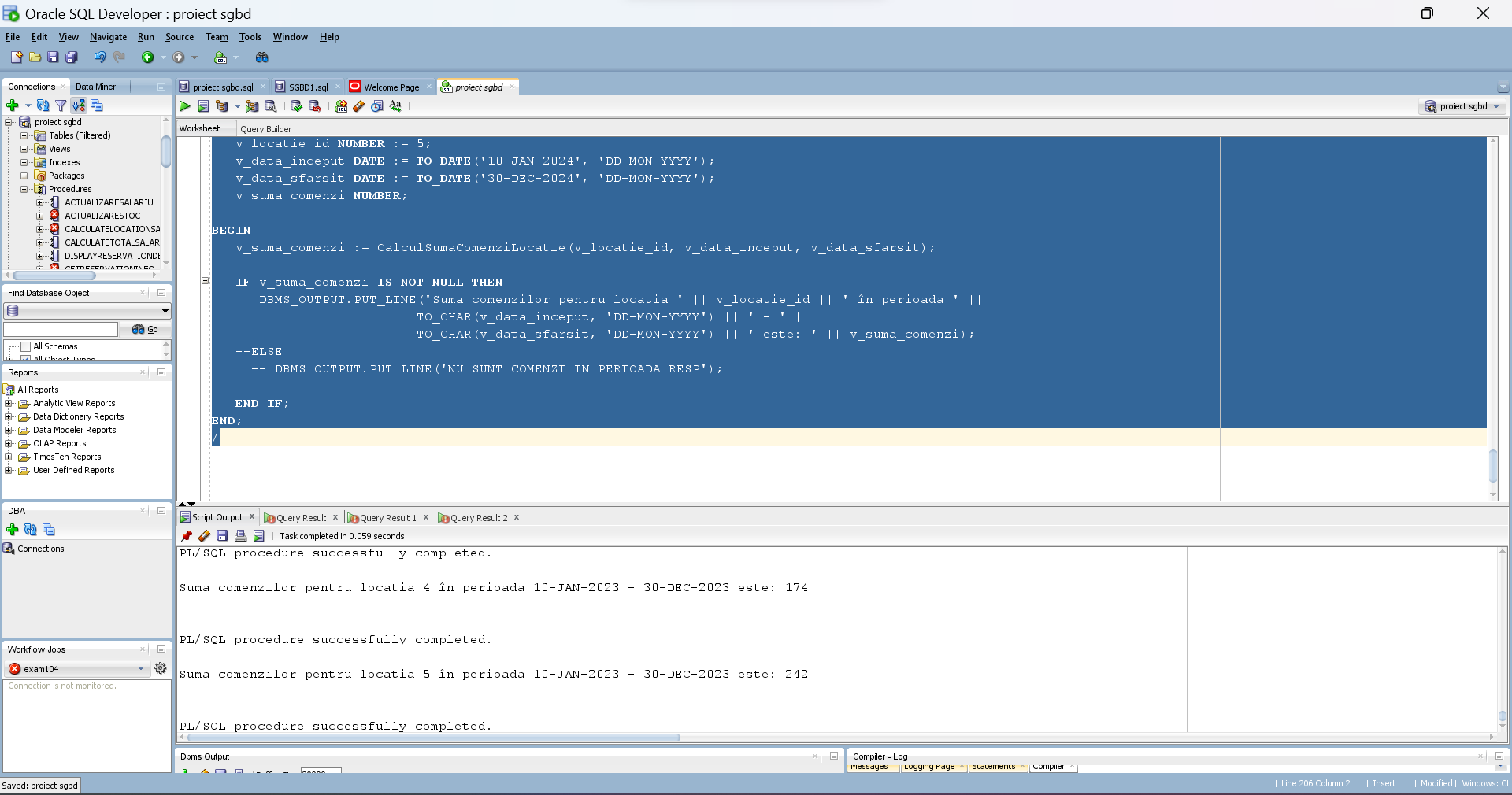
-- DBMS\_OUTPUT.PUT\_LINE('NU SUNT COMENZI IN PERIOADA RESP');

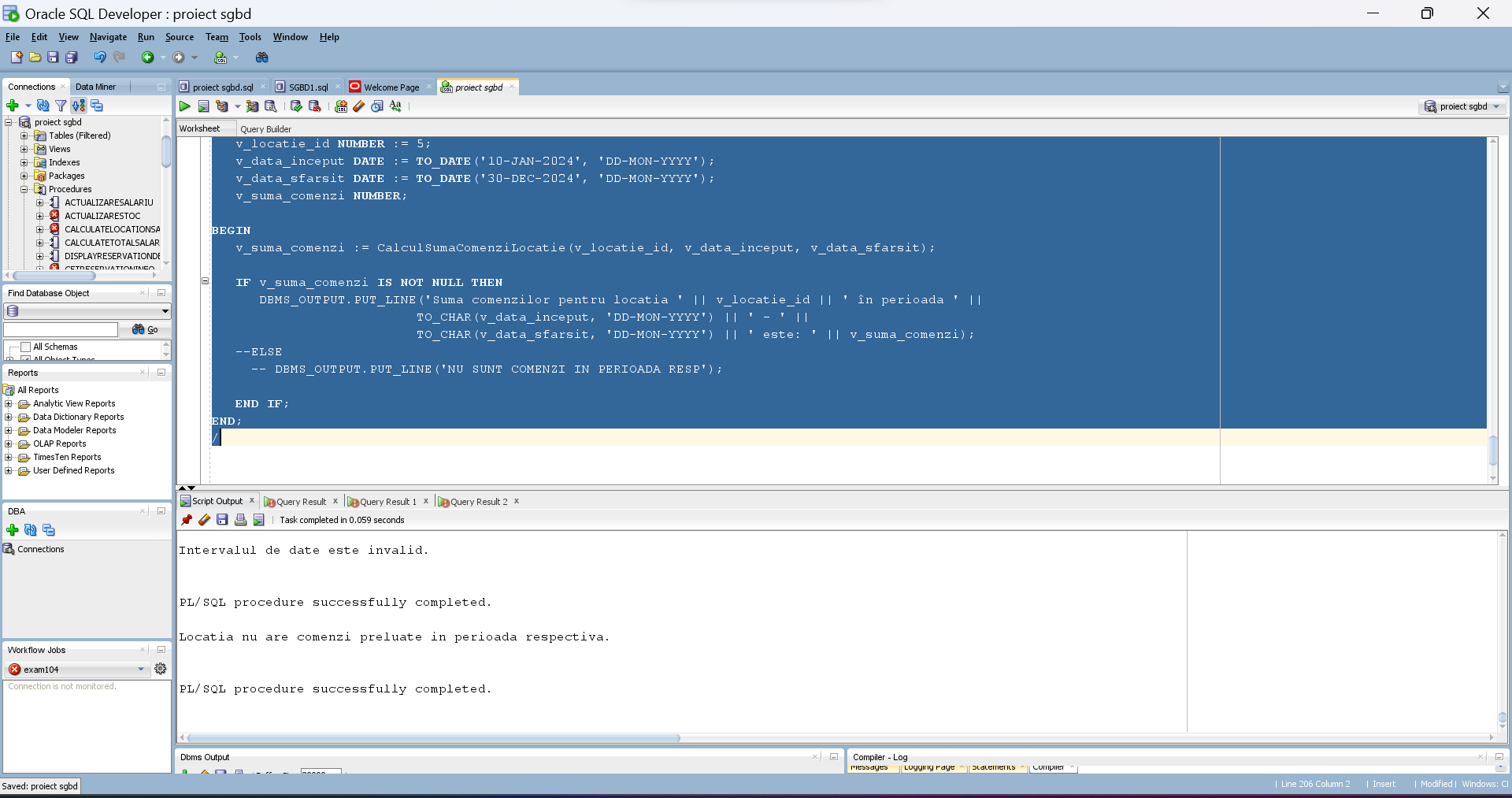
END IF;

END;

/







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

Să se facă update la salariul angajaților din fiecare locație, în funcție de numărul de comenzi care au fost procesate în locația respectivă într-o anumită perioadă de timp.

CREATE OR REPLACE PROCEDURE ActualizareSalariu (

locatie\_id\_param IN NUMBER,

data\_inceput\_param IN DATE,

data\_sfarsit\_param IN DATE

)

IS

v\_salariu\_bonus CONSTANT NUMBER := 250;

-- Exceptie pentru interval de date invalid

InvalidDateRange EXCEPTION;

-- Exceptie pentru prea multe randuri returnate

--TooManyRows EXCEPTION;

CURSOR c\_employee\_data IS

SELECT DISTINCT al.angajat\_id,

a.nume,

j.job\_titlu,

a.salariu,

COUNT(DISTINCT cl.comanda\_id) \* v\_salariu\_bonus + a.salariu AS salariu\_nou

FROM Angajat\_Loc al

JOIN Comanda\_Loc cl ON al.locatie\_id = cl.locatie\_id

JOIN Comanda c ON cl.comanda\_id = c.comanda\_id

JOIN Angajat a ON al.angajat\_id = a.angajat\_id

JOIN Job j ON a.job\_id = j.job\_id

WHERE al.locatie\_id = locatie\_id\_param

AND c.data\_comanda BETWEEN data\_inceput\_param AND data\_sfarsit\_param

GROUP BY al.angajat\_id, a.nume, j.job\_titlu, a.salariu;

v\_row\_count NUMBER := 0;

BEGIN

-- Verifica daca intervalul de date este valid

IF data\_inceput\_param > data\_sfarsit\_param THEN

RAISE InvalidDateRange;

END IF;

FOR emp\_data IN c\_employee\_data LOOP

v\_row\_count := v\_row\_count + 1;

IF v\_row\_count > 2 THEN

RAISE TOO\_MANY\_ROWS;

END IF;

-- Actualizeaza salariul

UPDATE Angajat

SET salariu = emp\_data.salariu\_nou

WHERE angajat\_id = emp\_data.angajat\_id;

END LOOP;

-- Verifica daca nu s-au gasit date

IF v\_row\_count = 0 THEN

RAISE NO\_DATA\_FOUND;

END IF;

DBMS\_OUTPUT.PUT\_LINE('Actualizare salarii finalizata cu succes.');

EXCEPTION

WHEN InvalidDateRange THEN

DBMS\_OUTPUT.PUT\_LINE('Intervalul de date este invalid.');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Prea multe randuri returnate.');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu au fost gasite date.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('O eroare neasteptata a aparut: ' || SQLERRM);

END ActualizareSalariu;

/

DECLARE

locatie\_id\_param NUMBER := 4;

data\_inceput\_param DATE := TO\_DATE('01-JUN-2024', 'DD-MON-YYYY');

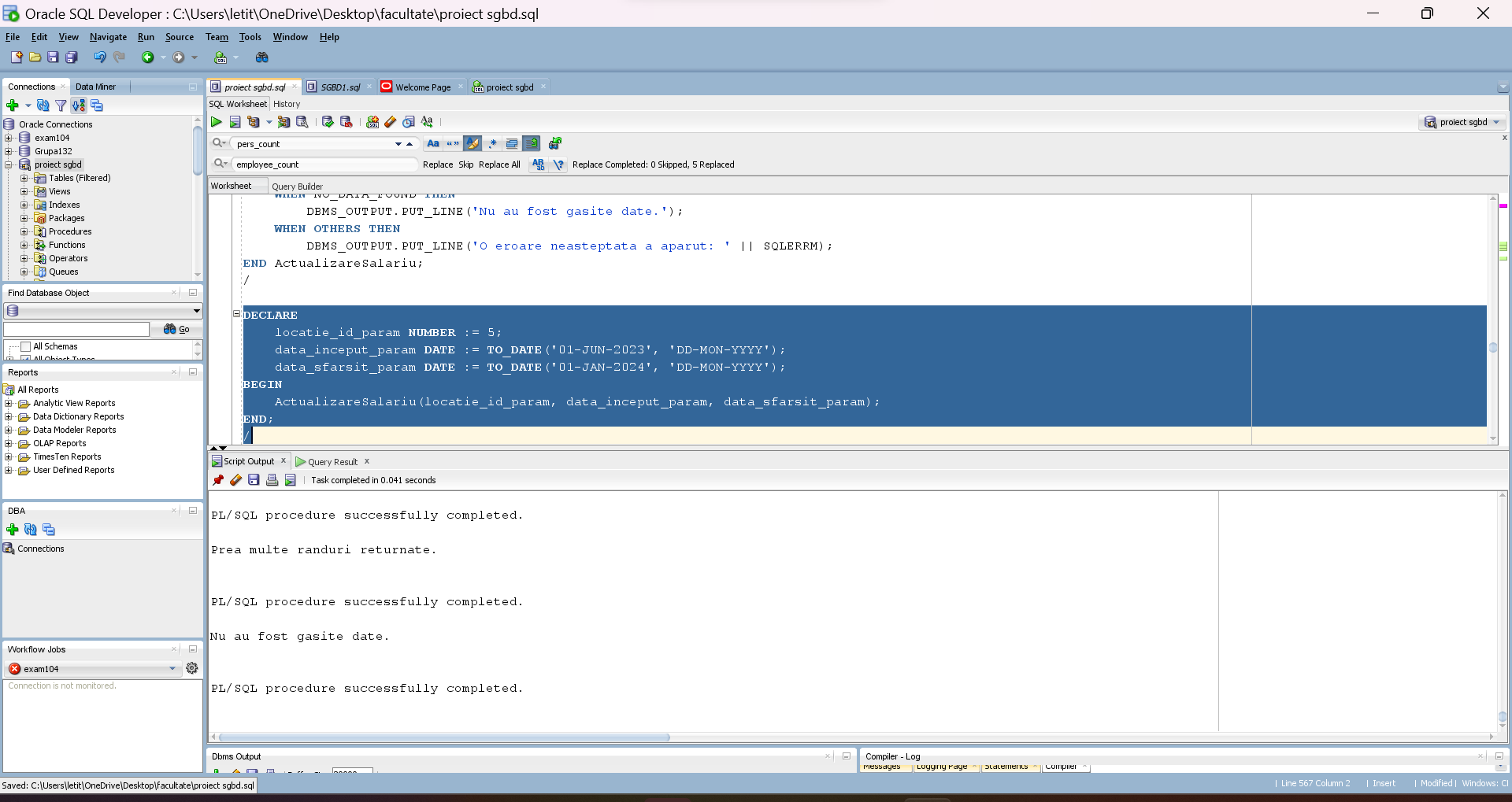
data\_sfarsit\_param DATE := TO\_DATE('01-JAN-2023', 'DD-MON-YYYY');

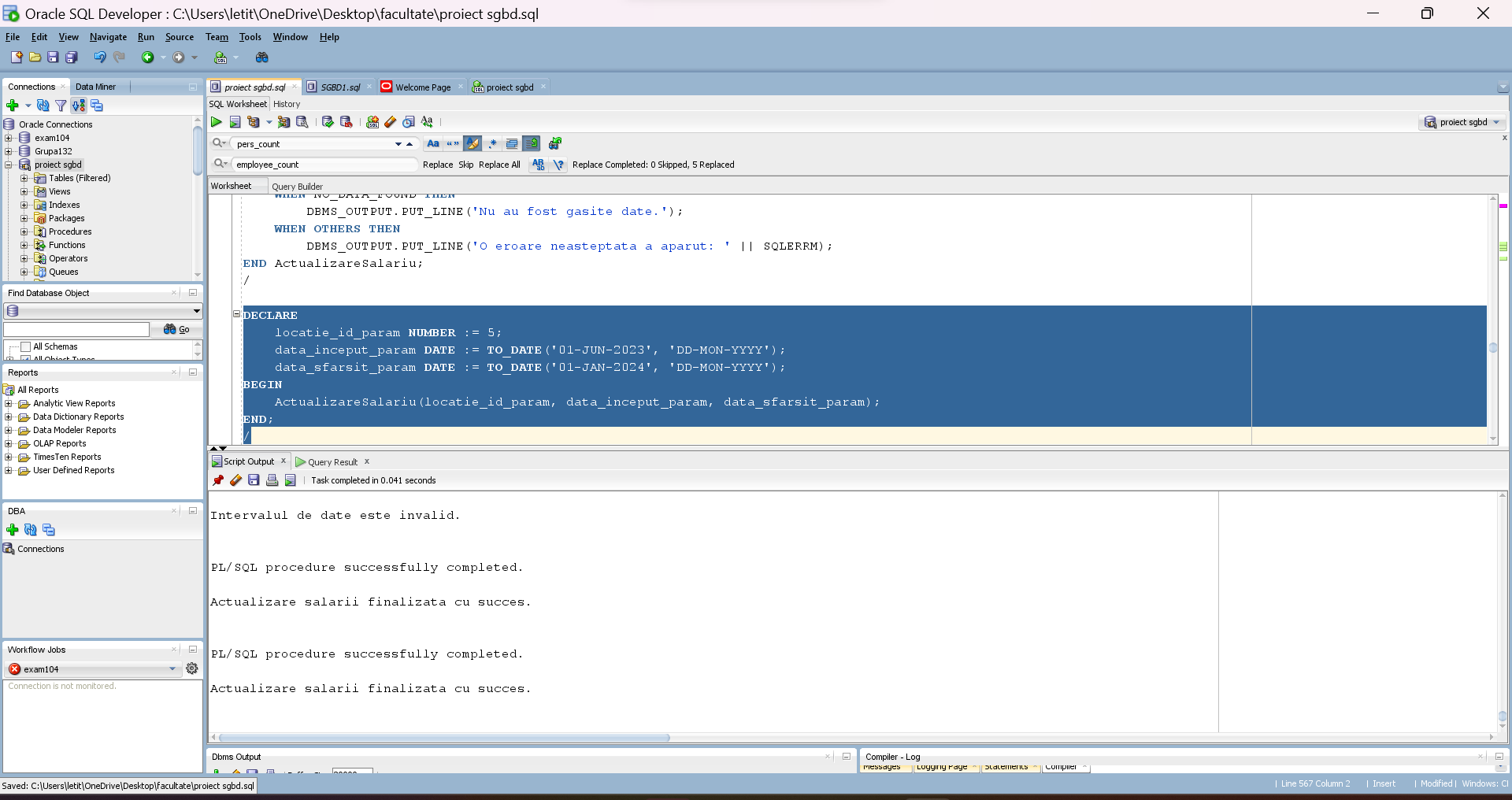
BEGIN

ActualizareSalariu(locatie\_id\_param, data\_inceput\_param, data\_sfarsit\_param);

END;

/





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

Definiți un declanșator care să permită realizarea unor modificari asupra tabelului Comanda (INSERT, UPDATE, DELETE) doar în intervalul de ore 8:00 – 14:00, de luni până sâmbătă.

CREATE OR REPLACE TRIGGER com\_trigger

BEFORE INSERT OR UPDATE OR DELETE ON Comanda

BEGIN

IF (TO\_CHAR(SYSDATE,'D') = 1)

OR (TO\_CHAR(SYSDATE,'HH24') NOT BETWEEN 8 AND 14)

THEN

RAISE\_APPLICATION\_ERROR(-20001,'tabelul nu poate fi actualizat');

END IF;

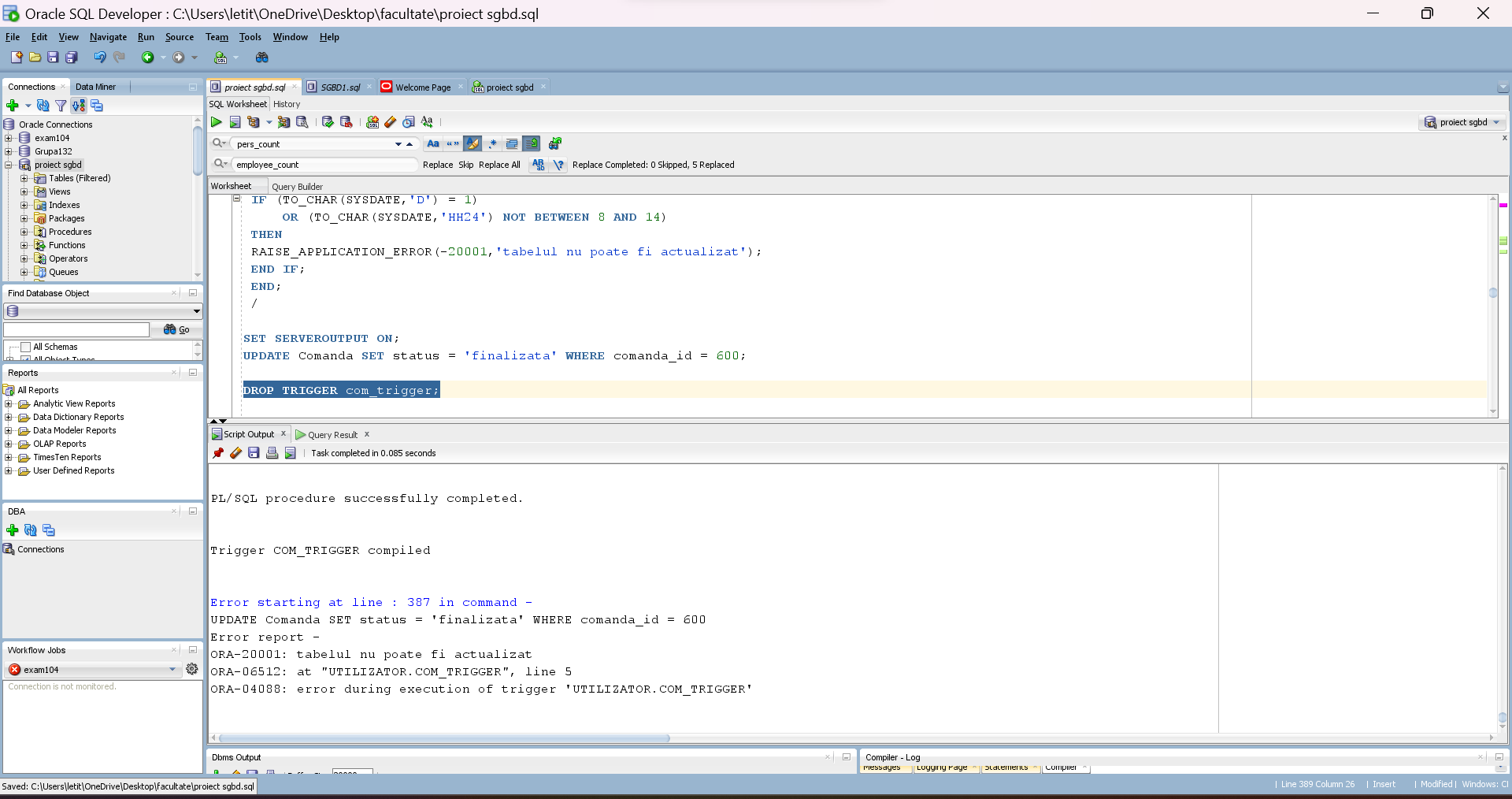
END;

/

SET SERVEROUTPUT ON;

UPDATE Comanda SET status = 'finalizata' WHERE comanda\_id = 600;

DROP TRIGGER com\_trigger;



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

Definiți un declanșator prin care să nu se permită mărirea salariilor angajaților din tabelul Angajat.

CREATE OR REPLACE TRIGGER sal\_trigger

BEFORE UPDATE OF salariu ON Angajat

FOR EACH ROW

BEGIN

IF (:NEW.salariu > :OLD.salariu) THEN

RAISE\_APPLICATION\_ERROR(-20002,'salariul nu poate fi marit');

END IF;

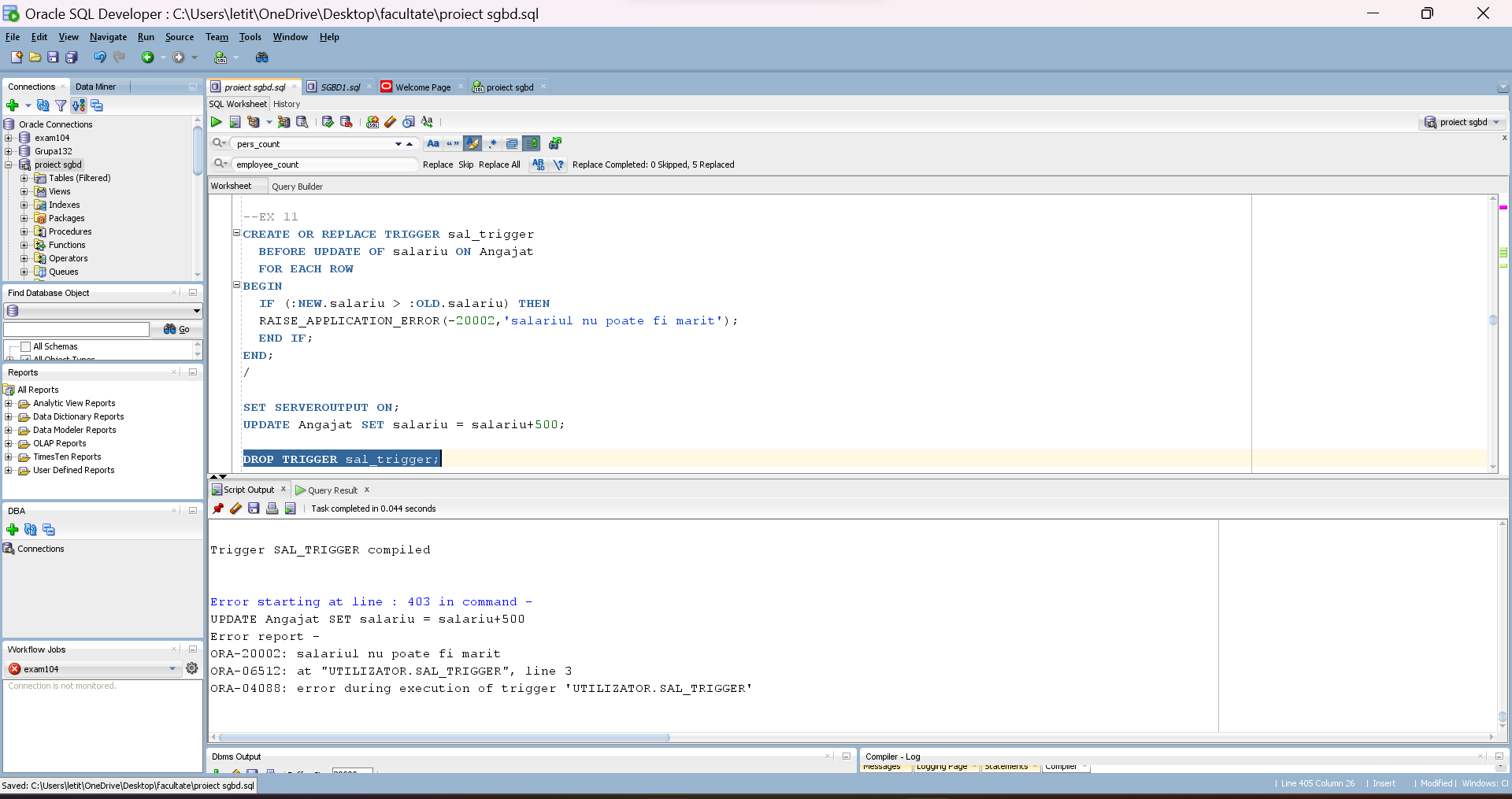
END;

/

SET SERVEROUTPUT ON;

UPDATE Angajat SET salariu = salariu+500;

DROP TRIGGER sal\_trigger;



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

Definiți un declanșator care să întroducă date în tabelul Baza după ce utilizatorul a folosit o comandă LDD .

CREATE TABLE Baza

(utilizator VARCHAR2(30),

nume\_bd VARCHAR2(50),

eveniment VARCHAR2(20),

nume\_obiect VARCHAR2(30),

data DATE);

CREATE OR REPLACE TRIGGER baz\_trigger

AFTER CREATE OR DROP OR ALTER ON SCHEMA

BEGIN

INSERT INTO Baza

VALUES (SYS.LOGIN\_USER, SYS.DATABASE\_NAME, SYS.SYSEVENT, SYS.DICTIONARY\_OBJ\_NAME, SYSDATE);

END;

/

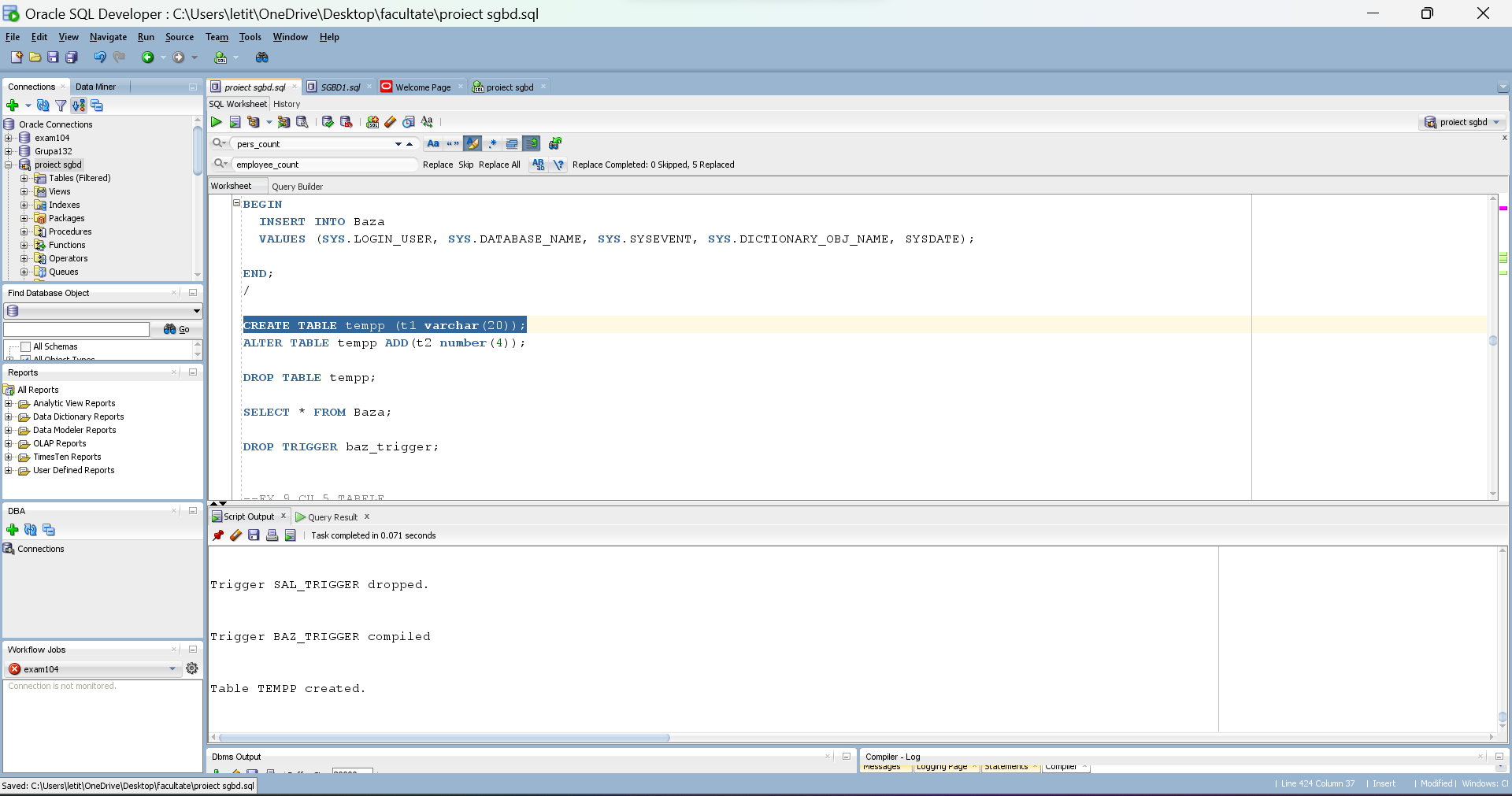
CREATE TABLE tempp (t1 varchar(20));

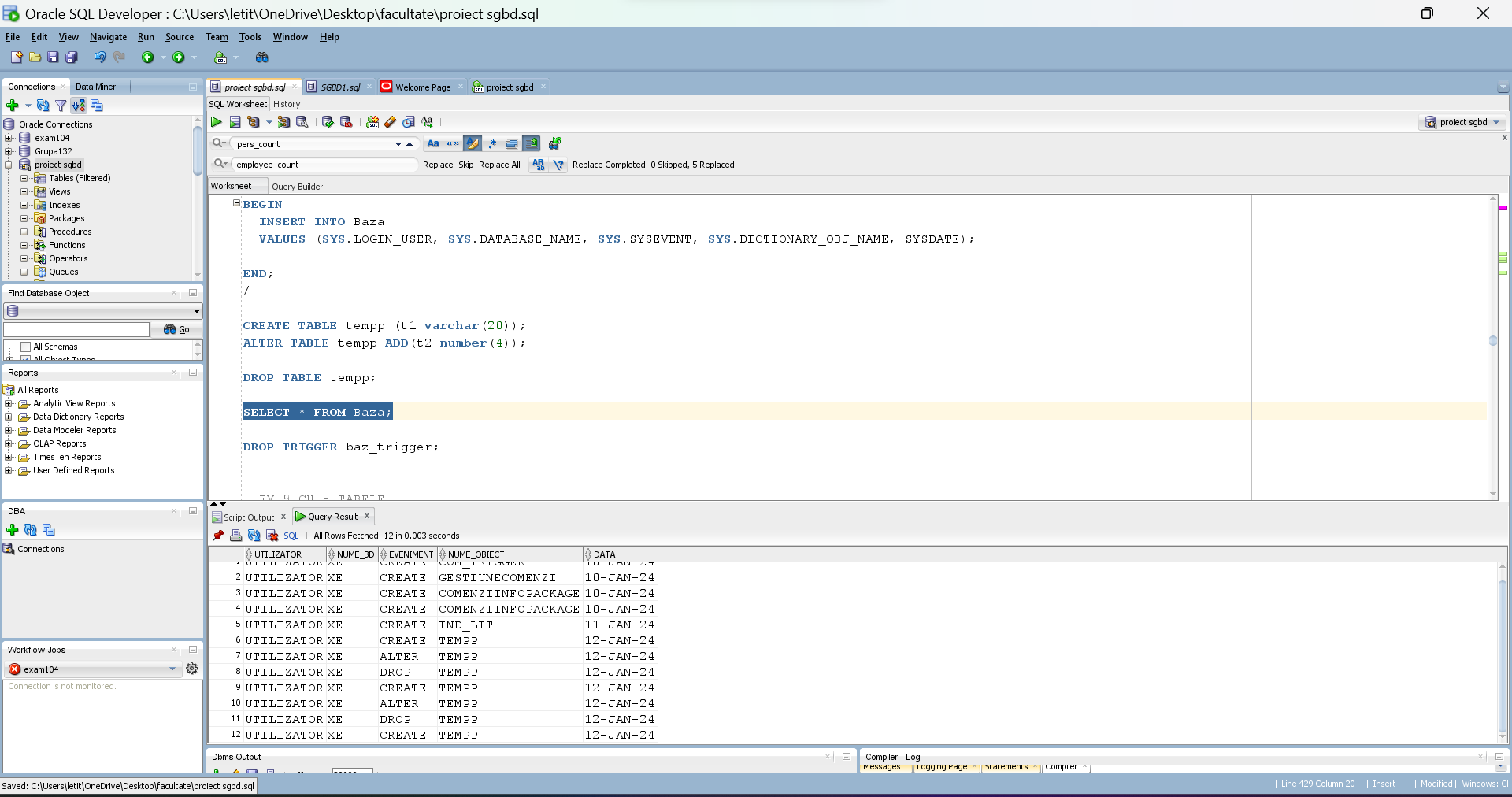
ALTER TABLE tempp ADD(t2 number(4));

DROP TABLE tempp;

SELECT \* FROM Baza;

DROP TRIGGER baz\_trigger;





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

CREATE OR REPLACE PACKAGE ProiectPackage AS

--EX 6

PROCEDURE DisplayReservationDetails(

location\_id\_param IN NUMBER

);

--EX 7

-- PROCEDURE CalculateTotalSalariesForEachLocation(

--);

--EX 8

FUNCTION CalculSumaComenziLocatie(

p\_locatie\_id IN NUMBER,

p\_data\_inceput IN DATE,

p\_data\_sfarsit IN DATE

)RETURN NUMBER;

--EX 9

PROCEDURE ActualizareSalariu (

locatie\_id\_param IN NUMBER,

data\_inceput\_param IN DATE,

data\_sfarsit\_param IN DATE

);

END ProiectPackage;

/

SHOW ERRORS;

CREATE OR REPLACE PACKAGE BODY ProiectPackage AS

--EX 6

PROCEDURE DisplayReservationDetails(

location\_id\_param IN NUMBER

)

IS

TYPE ReservationInfoType IS RECORD (

reservation\_date DATE,

employee\_count NUMBER,

client\_name VARCHAR2(50)

);

TYPE DateReservations IS TABLE OF DATE INDEX BY PLS\_INTEGER;

TYPE EmployeeCounts IS TABLE OF ReservationInfoType ;

TYPE ClientNames IS VARRAY(50) OF ReservationInfoType;

reservation\_dates DateReservations;

reservation\_employee\_counts EmployeeCounts;

reservation\_client\_names ClientNames;

BEGIN

reservation\_dates := DateReservations();

reservation\_employee\_counts := EmployeeCounts();

reservation\_client\_names := ClientNames();

FOR rec IN (

SELECT r.data\_rezervare, COUNT(DISTINCT al.angajat\_id) AS employee\_count, c.nume1 AS client\_name

FROM Rezervare r

JOIN Angajat\_Loc al ON r.locatie\_id = al.locatie\_id

JOIN Angajat a ON al.angajat\_id = a.angajat\_id

JOIN Client c ON r.client\_id = c.client\_id

WHERE r.locatie\_id = location\_id\_param

GROUP BY r.data\_rezervare, c.nume1

)

LOOP

reservation\_dates(reservation\_dates.COUNT + 1) := rec.data\_rezervare;

IF rec.employee\_count IS NOT NULL THEN

reservation\_employee\_counts.extend;

reservation\_employee\_counts(reservation\_employee\_counts.LAST). employee\_count := rec. employee\_count;

ELSE

reservation\_employee\_counts.extend;

reservation\_employee\_counts(reservation\_employee\_counts.LAST). employee\_count := 0;

END IF;

reservation\_client\_names.extend;

reservation\_client\_names(reservation\_client\_names.LAST).client\_name := rec.client\_name;

END LOOP;

FOR i IN reservation\_dates.FIRST..reservation\_dates.LAST

LOOP

DBMS\_OUTPUT.PUT\_LINE('Rezervare la data: ' || TO\_CHAR(reservation\_dates(i), 'DD-MON-YYYY'));

DBMS\_OUTPUT.PUT\_LINE('Numarul angajatilor care lucreaza în locatia respectiva: ' || reservation\_employee\_counts(i).employee\_count);

DBMS\_OUTPUT.PUT\_LINE('Numele clientului: ' || reservation\_client\_names(i).client\_name);

END LOOP;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu au fost gasite rezervari pentru locatia specificata.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare neasteptata: ' || SQLERRM);

END DisplayReservationDetails;

--EX 7

PROCEDURE CalculateTotalSalariesForEachLocation AS

--CURSOR LocationCursor IS

-- SELECT DISTINCT locatie\_id

-- FROM Angajat\_Loc;

CURSOR SalaryCursor (location\_id NUMBER) IS

SELECT salariu

FROM Angajat\_Loc al

JOIN Angajat a ON al.angajat\_id = a.angajat\_id

WHERE al.locatie\_id = location\_id;

total\_salary NUMBER;

BEGIN

FOR loc IN

(SELECT DISTINCT locatie\_id

FROM Angajat\_Loc

)LOOP

total\_salary := 0;

FOR sal IN SalaryCursor(loc.locatie\_id) LOOP

total\_salary := total\_salary + sal.salariu;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Suma salariilor pentru locatia ' || loc.locatie\_id || ': ' || total\_salary);

END LOOP;

END CalculateTotalSalariesForEachLocation;

--EX 8

FUNCTION CalculSumaComenziLocatie(

p\_locatie\_id IN NUMBER,

p\_data\_inceput IN DATE,

p\_data\_sfarsit IN DATE

) RETURN NUMBER IS

v\_suma NUMBER := 0;

v\_numar\_comenzi NUMBER := 0;

v\_row\_count NUMBER := 0;

-- Exceptii proprii

LocatieNotFound EXCEPTION;

InvalidDateRange EXCEPTION;

BEGIN

IF p\_data\_inceput > p\_data\_sfarsit THEN

RAISE InvalidDateRange;

END IF;

SELECT COUNT(\*)

INTO v\_numar\_comenzi

FROM Locatie l

JOIN Comanda\_Loc cl on l.locatie\_id = cl.locatie\_id

JOIN Comanda c ON cl.comanda\_id = c.comanda\_id

WHERE l.locatie\_id = p\_locatie\_id

AND c.data\_comanda BETWEEN p\_data\_inceput AND p\_data\_sfarsit;

IF v\_numar\_comenzi = 0 THEN

RAISE LocatieNotFound;

END IF;

SELECT SUM(dc.pret\_unitar \* dc.cantitate)

INTO v\_suma

FROM Comanda\_Loc cl

JOIN Comanda c ON cl.comanda\_id = c.comanda\_id

JOIN DetaliiComanda dc ON c.detaliucomanda\_id = dc.detaliucomanda\_id

WHERE cl.locatie\_id = p\_locatie\_id

AND c.data\_comanda BETWEEN p\_data\_inceput AND p\_data\_sfarsit;

RETURN v\_suma;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista date.');

WHEN InvalidDateRange THEN

DBMS\_OUTPUT.PUT\_LINE('Intervalul de date este invalid.');

RETURN NULL;

WHEN LocatieNotFound THEN

DBMS\_OUTPUT.PUT\_LINE('Locatia nu are comenzi preluate in perioada respectiva.');

RETURN NULL;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('O eroare neasteptata a aparut.');

RETURN NULL;

END CalculSumaComenziLocatie;

--EX 9

PROCEDURE ActualizareSalariu (

locatie\_id\_param IN NUMBER,

data\_inceput\_param IN DATE,

data\_sfarsit\_param IN DATE

)IS

v\_salariu\_bonus CONSTANT NUMBER := 250;

-- Exceptie pentru interval de date invalid

InvalidDateRange EXCEPTION;

-- Exceptie pentru prea multe randuri returnate

--TooManyRows EXCEPTION;

-- Cursor pentru a obtine datele necesare

CURSOR c\_employee\_data IS

SELECT DISTINCT al.angajat\_id,

a.nume,

j.job\_titlu,

a.salariu,

COUNT(DISTINCT cl.comanda\_id) \* v\_salariu\_bonus + a.salariu AS salariu\_nou

FROM Angajat\_Loc al

JOIN Comanda\_Loc cl ON al.locatie\_id = cl.locatie\_id

JOIN Comanda c ON cl.comanda\_id = c.comanda\_id

JOIN Angajat a ON al.angajat\_id = a.angajat\_id

JOIN Job j ON a.job\_id = j.job\_id

WHERE al.locatie\_id = locatie\_id\_param

AND c.data\_comanda BETWEEN data\_inceput\_param AND data\_sfarsit\_param

GROUP BY al.angajat\_id, a.nume, j.job\_titlu, a.salariu;

-- Variabila pentru a numara randurile returnate

v\_row\_count NUMBER := 0;

BEGIN

-- Verifica daca intervalul de date este valid

IF data\_inceput\_param > data\_sfarsit\_param THEN

RAISE InvalidDateRange;

END IF;

-- Executa cursorul si actualizeaza salariile

FOR emp\_data IN c\_employee\_data LOOP

v\_row\_count := v\_row\_count + 1;

-- Verifica daca s-au gasit prea multe randuri

IF v\_row\_count > 2 THEN

RAISE TOO\_MANY\_ROWS;

END IF;

-- Actualizeaza salariul

UPDATE Angajat

SET salariu = emp\_data.salariu\_nou

WHERE angajat\_id = emp\_data.angajat\_id;

END LOOP;

-- Verifica daca nu s-au gasit date

IF v\_row\_count = 0 THEN

RAISE NO\_DATA\_FOUND;

END IF;

DBMS\_OUTPUT.PUT\_LINE('Actualizare salarii finalizata cu succes.');

EXCEPTION

WHEN InvalidDateRange THEN

DBMS\_OUTPUT.PUT\_LINE('Intervalul de date este invalid.');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Prea multe randuri returnate.');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu au fost gasite date.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('O eroare neasteptata a aparut: ' || SQLERRM);

END ActualizareSalariu;

END ProiectPackage;

/

