Sala Fitness

Proiect SGBD

Realizat de Rizea Mihai-Marius

Grupa 241

An Universitar 2024-2025

Cuprins

1. [Prezentați pe scurt baza de date (utilitatea ei).](#igkxha2zhc2) 5

⮚ **Tema** 5

⮚ **Descrierea modelului real** 5

⮚ **Utilitatea** 6

⮚ **Funcționalități** 6

⮚ **Infrastructura utilizată pentru implementarea proiectului:** 7

2. [Realizați diagrama entitate-relație (ERD): entitățile, relațiile și atributele trebuie definite în limba română (vezi curs SGBD, model de diagrama entitate-relație; nu se va accepta alt format).](#tuh84vwxtng4) 8

3. [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.](#hu884rqxjs4r) 9

4. [Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, adăugând toate constrângerile de integritate necesare (chei primare, cheile externe etc).](#bb3a72o2bvl) 13

[1.Tabela **SALA**:](#_1b2wsoouz8o1) 15

[2. Tabela **PRODUSE**:](#_uvfd13k2vrn) 15

[3. Tabela **APARATE:**](#_2xqn096kp5uj) 16

[4. Tabela **ABONAMENT**:](#_gy41697sb7k9) 17

[5. Tabela **CLIENT**:](#_uzmdba18w3li) 18

[6. Tabela **ACHIZITIE**:](#_i0eulc1wo6fk) 19

[7. Tabela **DETALII\_ACHIZITIE**:](#_x9xmhr1q8o8s) 20

[8. Tabela **DETINE**:](#_k2t7qzxw5c0x) 21

[9. Tabela **UTILITATI:**](#_h7ro41fkqxzn) 22

[10. Tabela **FACTURI**:](#_cvyccn13hfab) 23

[11. Tabela **FACTURI\_UTILITATI:**](#_v7djooikqlyi) 24

[12. Tabela **UTILITATI\_PLATITE\_PRIN:**](#_2yysqz1za9jr) 25

[13. Tabela **FACTURI\_ABONAMENT:**](#_sh6nmxwlktxh) 26

[14. Tabela **FACTURI\_PRODUSE:**](#_t7jm23lw5lir) 27

[5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru fiecare tabelă asociativă).](#k7p7g7oxkr20) 29

[1.Tabela **SALA**:](#2c2v63bikcy2) 29

[2. Tabela **PRODUSE**:](#g0h8bm1ohvb2) 29

[3. Tabela **APARATE:**](#9pipxj82dh33) 31

[4. Tabela **ABONAMENT**:](#71bzhrcaweak) 34

[5. Tabela **CLIENT**:](#rz3ptbvqeaje) 36

[6. Tabela **ACHIZITIE**:](#k7trgzk2ut4y) 39

[7. Tabela **DETALII\_ACHIZITIE**:](#sciut75oi1ps) 41

[8. Tabela **DETINE**:](#xyqzheqq4qor) 43

[9. Tabela **UTILITATI:**](#uc8nc02o5lxr) 45

[10. Tabela **FACTURI**:](#e85ak88ci5vy) 47

[11. Tabela **FACTURI\_UTILITATI:**](#ha66qvs5db3w) 51

[12. Tabela **UTILITATI\_PLATITE\_PRIN:**](#pvofdl14v5np) 53

[13. Tabela **FACTURI\_ABONAMENT:**](#wqz0ygk7chj) 55

[14. Tabela **FACTURI\_PRODUSE:**](#y6ypymsz01gz)  57

[6. 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.](#ltk8fjcjkxv8) 63

[1.Tabel Imbricat:](#3j0h85me0i25) 63

[2. Tabel Indexat:](#rnlisc9bl3nb) 68

[3. Vector**:**](#pxd33o35qv0x) 72

[7. 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.](#mdd8g4mv1gp0) 75

[1.Varianta cu ciclu cursor:](#4ffbrjgj8rnf) 75

[2. Varianta cu OPEN + Fetch:](#mms2o5snna6c) 77

[8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele create. Tratați toate excepțiile care pot apărea, incluzând excepțiile predefinite NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.](#1yhlk6b865bi) 80

[9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să aibă minim 2 parametri și să utilizeze într-o singură comandă SQL 5 dintre tabelele create. Definiți minim 2 excepții proprii, altele decât cele predefinite la nivel de sistem. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate.](#6o5ow0bs7v68) 93

[10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.](#qep8ee2g26wa) 100

[11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.](#oszdrpo7ax88) 102

[12. Definiți un trigger de tip LDD. Declanșați trigger-ul.](#xw7xodjqo2oa)  110

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

⦁ **Tema**

Tema acestui proiect este o bază de date realizată pentru o aplicație mobilă denumită "Fitness World", care are scopul de a monitoriza activitățile și echipamentele dintr-o sală de fitness și de a reține detalii esențiale despre acestea. Aplicația va permite utilizatorilor să urmărească disponibilitatea aparatelor, produsele disponibile, gestionarea sălii și activitatea membrilor.

⦁ **Descrierea Modelului Real**

Această bază de date este concepută pentru a gestiona eficient activitățile și echipamentele dintr-o sală de fitness. Ea permite urmărirea disponibilității aparatelor, gestionarea produselor disponibile, monitorizarea abonaților și administrarea achizițiilor și facturilor asociate.

Entitatea Sala stochează informații despre sălile de fitness, incluzând identificatorul sălii, numărul de clienți, veniturile generate, numărul de produse, numele managerului și starea de deschidere. Fiecare sală este legată de aparatele, produsele, utilitățile, facturile și abonamentele aferente.

Entitatea Aparate conține detalii despre echipamentele din sală, cum ar fi identificatorul aparatului, numele aparatului, masa maximă suportată, disponibilitatea, grupa musculară lucrată..

Entitatea Utilitati stochează informații despre utilitățile necesare pentru funcționarea sălii de fitness, incluzând identificatorul utilității, numele, cantitatea și prețul acesteia. Fiecare utilitate este legată de facturile de utilități.

Entitatea Abonament conține detalii despre abonamentele oferite de sală, incluzând identificatorul abonamentului, prețul, tipul abonamentului și data de valabilitate. Fiecare abonament este asociat cu o sală și un client.

Entitatea Produse stochează informații despre produsele disponibile în sală de fitness, incluzând identificatorul produsului, numele, cantitatea și prețul acestuia. Fiecare produs este legat de detalii de achiziție.

Entitatea Detalii\_Achizitie oferă informații detaliate despre fiecare achiziție, specificând produsele sau serviciile incluse în achiziție, inclusiv cantitatea.

Entitatea Achizitie înregistrează toate achizițiile efectuate în sală de fitness, incluzând detalii despre data achiziției. Fiecare achiziție este legată de detalii de achiziție.

Entitatea Client gestionează informații despre membrii sălii, cum ar fi identificatorul clientului, numele, prenumele, emailul și numărul de telefon. Fiecare client este legat de achiziții și abonamente.

Entitatea Facturi gestionează toate facturile generate, având trei subunități: Facturi\_Utilitati pentru facturile de utilități, Facturi\_Abonament pentru facturile de abonament și Facturi\_Produse pentru facturile de produse achiziționate. Toate facturile includ atributele de dată și sumă.

Aceste entități sunt interconectate pentru a oferi o imagine completă și detaliată asupra gestionării unei săli de fitness, permițând monitorizarea eficientă a echipamentelor, produselor, abonamentelor și activităților financiare.

⦁ **Utilitatea**

Utilitatea acestei baze de date este de a îmbunătăți gestionarea activităților și resurselor dintr-o sală de fitness, oferind acces rapid și ușor la informații detaliate despre aparate, produse, utilități, abonamente și clienți. Acest sistem ajută la gestionarea eficientă a operațiunilor zilnice și a costurilor asociate.

Informațiile stocate în baza de date pot fi utilizate pentru a planifica reviziile și întreținerea aparatelor, pentru a gestiona stocurile de produse și pentru a urmări consumul de utilități. De asemenea, baza de date permite monitorizarea abonamentelor membrilor și generarea de rapoarte detaliate despre activitățile și veniturile sălii.

Aceste rapoarte pot ajuta operatorii să optimizeze costurile, să îmbunătățească serviciile oferite și să ia decizii informate pentru creșterea afacerii. În plus, istoricul de întreținere și utilizare al aparatelor și produselor poate fi util în cazul în care sala dorește să îmbunătățească echipamentele sau să ajusteze ofertele de servicii pentru a satisface mai bine nevoile clienților.

⦁ **Funcționalități**

Funcționalitățile acestei baze de date includ stocarea și gestionarea informațiilor despre sălile de fitness, aparate, produse, utilități, abonamente și clienți. De asemenea, aceasta permite adăugarea și actualizarea informațiilor despre săli, precum și generarea de acestora în ceea ce privește finanțele și utilitățile.

Prin urmare, această bază de date este foarte utilă pentru utilizatorii care dețin săli de fitness sau a clienților acestora oferindu-le acces rapid la informații. Acest lucru poate ajuta la planificarea reviziilor și a altor activități de întreținere, menținând aparatele în condiții optime și reducând costurile generale de funcționare, in plus pentru clienti, îi ajută să vadă ce produse și abonamente sunt disponibile intr-un mod mult mai confortabil.

⦁ **Infrastructura utilizată pentru implementarea proiectului:**

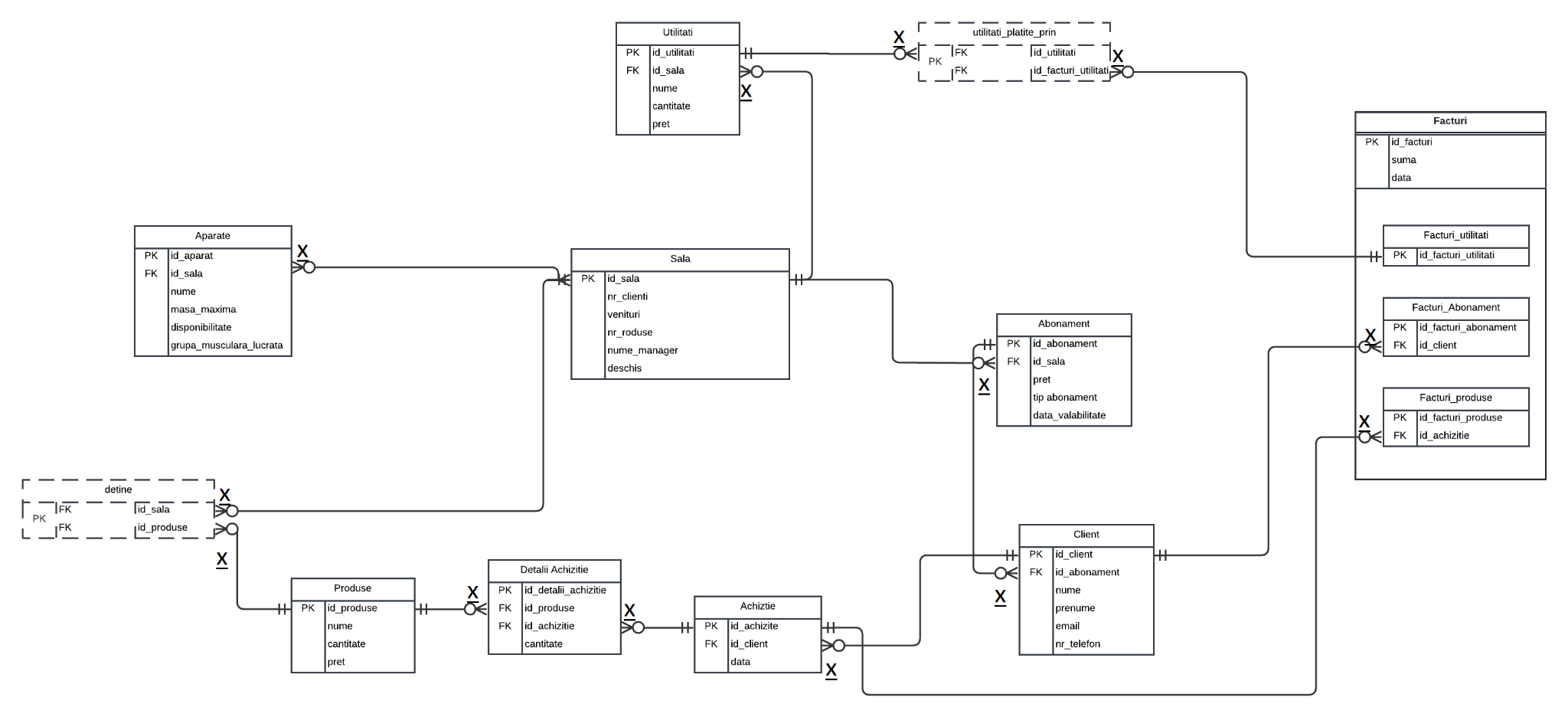
Oracle SQL Developer Versiunea 23.3.0.284 Build 284.2209

Configuratia hardware este de 16 GB RAM, iar sistemul de operare este Windows 10, nu a fost folosită o masina virtuala pentru efectuarea proiectului.

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



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



Regulile și constrângerile din baza de date descrisă sunt:

1.Tabela SALA:

⦁ id\_sala este cheia primară și trebuie să fie unică pentru fiecare înregistrare din tabel.

⦁ nr\_clienti trebuie să fie un număr întreg pozitiv.

⦁ venituri trebuie să fie o valoare numerică pozitivă.

⦁ nr\_produse trebuie să fie un număr întreg pozitiv.

⦁ nume\_manager trebuie să aibă o lungime maximă de 255 de caractere.

⦁ deschis trebuie să aibă o lungime maximă de 255 de caractere (va contine valorile: ‘da’/’nu’).

2. Tabela APARATE:

⦁ id\_aparat este cheia primară și trebuie să fie unică pentru fiecare înregistrare din tabel.

⦁ id\_sala este o cheie externă către tabela SALA și trebuie să se refere la un id\_sala existent în acea tabelă.

⦁ nume trebuie să aibă o lungime maximă de 100 de caractere.

⦁ masa\_maxima trebuie să fie un număr întreg pozitiv.

⦁ disponibilitate trebuie să aibă o lungime maximă de 255 de caractere.

⦁ grupa\_musculara\_lucrata trebuie să aibă o lungime maximă de 255 de caractere.

3. Tabela UTILITATI:

⦁ id\_utilitati este cheia primară și trebuie să fie unică pentru fiecare înregistrare din tabel.

⦁ id\_sala este o cheie externă către tabela SALA și trebuie să se refere la un id\_sala existent în acea tabelă.

⦁ nume trebuie să aibă o lungime maximă de 100 de caractere.

⦁ cantitate trebuie să fie o valoare numerică pozitivă.

⦁ pret trebuie să fie o valoare numerică pozitivă.

4. Tabela PRODUSE:

⦁ id\_produse este cheia primară și trebuie să fie unică pentru fiecare înregistrare din tabel.

⦁ nume trebuie să aibă o lungime maximă de 100 de caractere.

⦁ cantitate trebuie să fie o valoare numerică pozitivă.

⦁ pret trebuie să fie o valoare numerică pozitivă.

5. Tabela DETALII\_ACHIZITIE:

⦁ id\_detalii\_achizitie este cheia primară și trebuie să fie unică pentru fiecare înregistrare din tabel.

⦁ id\_produse este o cheie externă către tabela PRODUSE și trebuie să se refere la un id\_produse existent în acea tabelă.

⦁ id\_achizitie este o cheie externă către tabela ACHIZITIE și trebuie să se refere la un id\_achizitie existent în acea tabelă.

⦁ cantitate trebuie să fie o valoare numerică pozitivă.

6. Tabela ACHIZITIE:

⦁ id\_achizitie este cheia primară și trebuie să fie unică pentru fiecare înregistrare din tabel.

⦁ id\_client este o cheie externă către tabela CLIENT și trebuie să se refere la un id\_client existent în acea tabelă.

⦁ data trebuie să fie de tip DATE.

7. Tabela CLIENT:

⦁ id\_client este cheia primară și trebuie să fie unică pentru fiecare înregistrare din tabel.

⦁ id\_abonament este o cheie externă către tabela ABONAMENT și trebuie să se refere la un id\_abonament existent în acea tabelă

⦁ nume trebuie să aibă o lungime maximă de 255 de caractere.

⦁ prenume trebuie să aibă o lungime maximă de 255 de caractere.

⦁ email trebuie să aibă o lungime maximă de 255 de caractere.

⦁ nr\_telefon trebuie să aibă o lungime maximă de 15 caractere.

8. Tabela ABONAMENT:

⦁ id\_abonament este cheia primară și trebuie să fie unică pentru fiecare înregistrare din tabel.

⦁ id\_sala este o cheie externă către tabela SALA și trebuie să se refere la un id\_sala existent în acea tabelă.

⦁ pret trebuie să fie o valoare numerică pozitivă.

⦁ tip\_abonament trebuie să aibă o lungime maximă de 100 de caractere.

⦁ data\_valabilitate trebuie să fie de tip DATE.

9. Tabela FACTURI:

⦁ id\_facturi este cheia primară și trebuie să fie unică pentru fiecare înregistrare din tabel.

⦁ data trebuie să fie de tip DATE.

⦁ suma trebuie să fie o valoare numerică pozitivă.

10. Tabela FACTURI\_UTILITATI:

⦁ id\_facturi\_utilitati + id\_facturi este cheia primară compusă și trebuie să fie unică pentru fiecare înregistrare din tabel.

11. Tabela FACTURI\_ABONAMENT:

⦁ id\_facturi\_abonament + id\_facturi este cheia primară compusă și trebuie să fie unică pentru fiecare înregistrare din tabel.

⦁ id\_client este o cheie externă către tabela CLIENT și trebuie să se refere la un id\_client existent în acea tabelă.

12. Tabela FACTURI\_PRODUSE:

⦁ id\_facturi\_produse + id\_facturi este cheia primară compusă și trebuie să fie unică pentru fiecare înregistrare din tabel.

⦁ id\_achizitie este o cheie externă către tabela ACHIZITIE și trebuie să se refere la un id\_achizitie existent în acea tabelă.

13. Tabela DETINE:

⦁ id\_sala și id\_produse formează împreună cheia primară și trebuie să fie unică pentru fiecare înregistrare din tabel.

⦁ id\_sala este o cheie externă către tabela SALA și trebuie să se refere la un id\_sala existent în acea tabelă.

⦁ id\_produse este o cheie externă către tabela PRODUSE și trebuie să se refere la un id\_produse existent în acea tabelă.

14. Tabela UTILITATI\_PLATITE\_PRIN:

⦁ id\_utilitati și id\_facturi\_utilitati formează împreună cheia primară și trebuie să fie unică pentru fiecare înregistrare din tabel.

⦁ id\_utilitati este o cheie externă către tabela UTILITATI și trebuie să se refere la un id\_utilitati existent în acea tabelă.

⦁ id\_facturi\_utilitati este o cheie externă către tabela FACTURI\_UTILITATI și trebuie să se refere la un id\_facturi\_utilitati existent în acea tabelă.

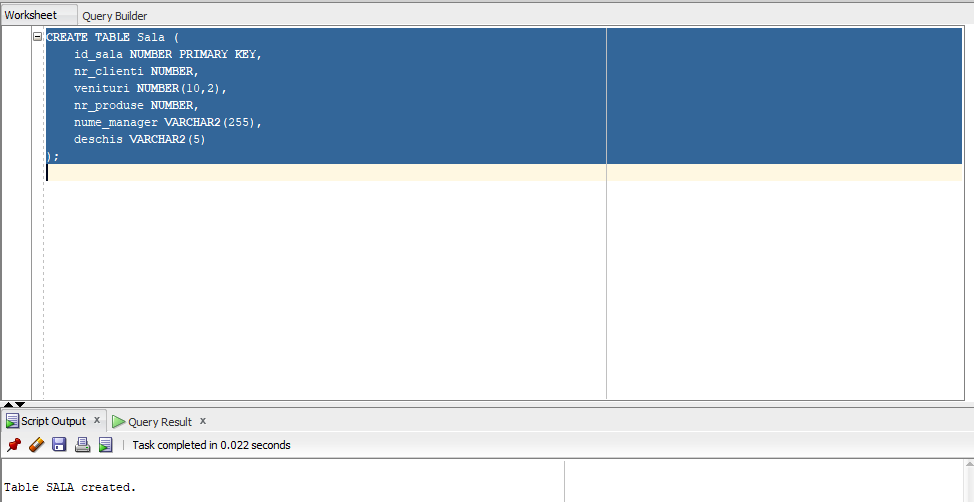
4.Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, adăugând toate constrângerile de integritate necesare (chei primare, cheile externe etc).

* **Sala - Aparate**: Sala are o relație de tip "one-to-many" cu Aparate, deoarece o sală poate avea mai multe aparate (sau niciunul), dar fiecare aparat este asociat doar unei săli. Cardinalitatea este 1:M(0).
* **Sala - Utilitati**: Sala are o relație de tip "one-to-many" cu Utilitati, deoarece o sală poate utiliza mai multe utilități (sau niciuna), dar fiecare utilitate este asociată doar unei săli. Cardinalitatea este 1:M(0).
* **Sala - Produse**: Sala are o relație de tip "many-to-many" cu Produse, deoarece o sală poate avea mai multe produse (sau niciunul), iar un produs poate fi asociat mai multor săli (sau la niciuna). Această relație este implementată prin intermediul unei tabele intermediare, care este detine. Cardinalitatea este M(0):M(0).
* **Sala - Abonament**: Sala are o relație de tip "one-to-many" cu Abonament, deoarece o sală poate avea mai multe abonamente (sau niciunul), dar fiecare abonament este asociat doar unei săli. Cardinalitatea este 1:M(0).
* **Achizitie - Detalii\_Achizitie**: Achizitie are o relație de tip "one-to-many" cu Detalii\_Achizitie, deoarece o achiziție poate avea mai multe detalii de achiziție (sau niciuna), dar fiecare detaliu de achiziție este asociat doar unei achiziții. Cardinalitatea este 1:M(0).
* **Achizitie - Client**: Achizitie are o relație de tip "many-to-one" cu Client, deoarece un client poate face mai multe achiziții (sau niciuna), dar fiecare achiziție este asociată doar unui singur client. Cardinalitatea este M(0):1.
* **Client - Abonament**: Client are o relație de tip "one-to-many" cu Abonament, deoarece un client poate avea mai multe abonamente (sau niciunul), dar fiecare abonament este asociat doar unui singur client. Cardinalitatea este 1:M(0).
* **Client - Facturi\_Abonament**: Client are o relație de tip "one-to-many" cu Facturi\_Abonament, deoarece un client poate primi mai multe facturi pentru abonament (sau niciuna), dar fiecare factură pentru abonament este asociată doar unui singur client. Cardinalitatea este 1:M(0).
* **Utilitati - Facturi\_Utilitati**: Utilitati și Facturi\_Utilitati au o relație de tip "many-to-many", deoarece o utilitate poate fi asociată cu mai multe facturi pentru utilități și, invers, o factură pentru utilități poate fi asociată cu mai multe utilități. Această relație este implementată prin intermediul unei tabele intermediare, care este Utilitati\_Plătite\_Prin. Astfel, cardinalitatea este M(1):M(1)
* **Produse - Facturi\_Produse**: Produse au o relație de tip "one-to-many" cu Facturi\_Produse, deoarece un produs poate fi inclus în mai multe facturi pentru produse (sau niciuna), dar fiecare factură pentru produse este asociată doar unui singur produs. Cardinalitatea este 1:M(0).
* **Facturi - Facturi\_Utilitati**: Facturi au o relație de tip "one-to-many" cu Facturi\_Utilitati, deoarece o factură poate fi asociată cu mai multe facturi pentru utilități (sau niciuna), dar fiecare factură pentru utilități este asociată doar unei singure facturi principale. Cardinalitatea este 1:M(0).
* **Facturi - Facturi\_Produse**: Facturi au o relație de tip "one-to-many" cu Facturi\_Produse, deoarece o factură poate fi asociată cu mai multe facturi pentru produse (sau niciuna), dar fiecare factură pentru produse este asociată doar unei singure facturi principale. Cardinalitatea este 1:M(0).
* **Facturi - Facturi\_Abonament**: Facturi au o relație de tip "one-to-many" cu Facturi\_Abonament, deoarece o factură poate fi asociată cu mai multe facturi pentru abonament (sau niciuna), dar fiecare factură pentru abonament este asociată doar unei singure facturi principale. Cardinalitatea este 1:M(0).
* Codul SQL:

## --1.Pentru tabelul SALA

| CREATE TABLE Sala (  id\_sala NUMBER PRIMARY KEY,  nr\_clienti NUMBER,  venituri NUMBER(10,2),  nr\_produse NUMBER,  nume\_manager VARCHAR2(255),  deschis VARCHAR2(5)  ); |
| --- |

Screenshot:

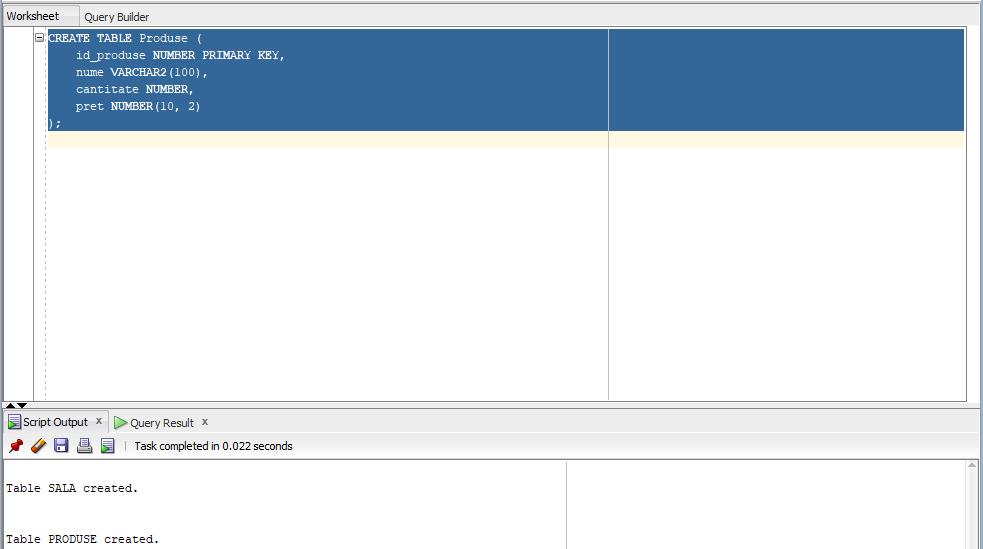


## 

## --2.Pentru tabelul PRODUSE

| CREATE TABLE Produse (  id\_produse NUMBER PRIMARY KEY,  nume VARCHAR2(100),  cantitate NUMBER,  pret NUMBER(10, 2)  ); |
| --- |

Screenshot:



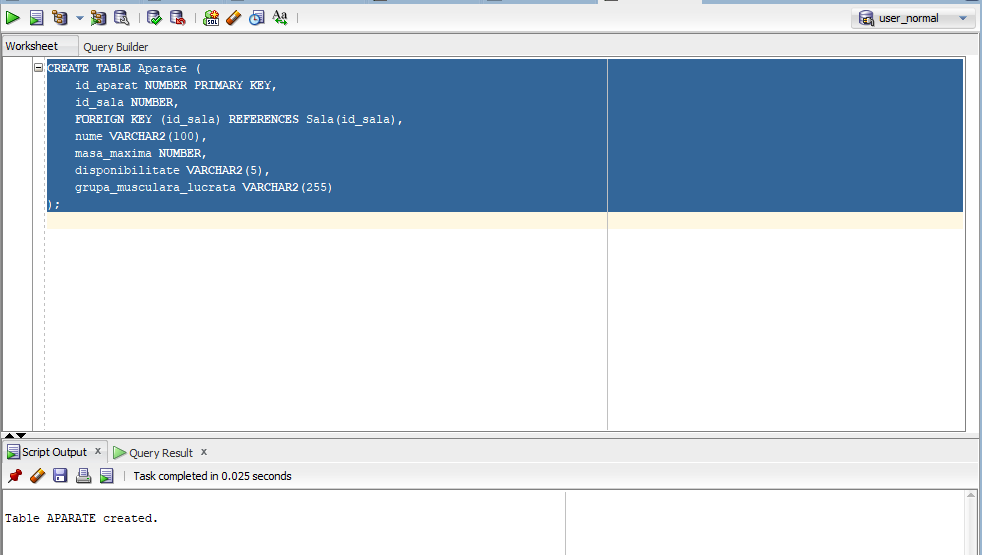
## 

## 

## --3.Pentru tabelul APARATE

| CREATE TABLE Aparate (  id\_aparat NUMBER PRIMARY KEY,  id\_sala NUMBER,  FOREIGN KEY (id\_sala) REFERENCES Sala(id\_sala),  nume VARCHAR2(100),  masa\_maxima NUMBER,  disponibilitate VARCHAR2(5),  grupa\_musculara\_lucrata VARCHAR2(255)  ); |
| --- |

Screenshot:

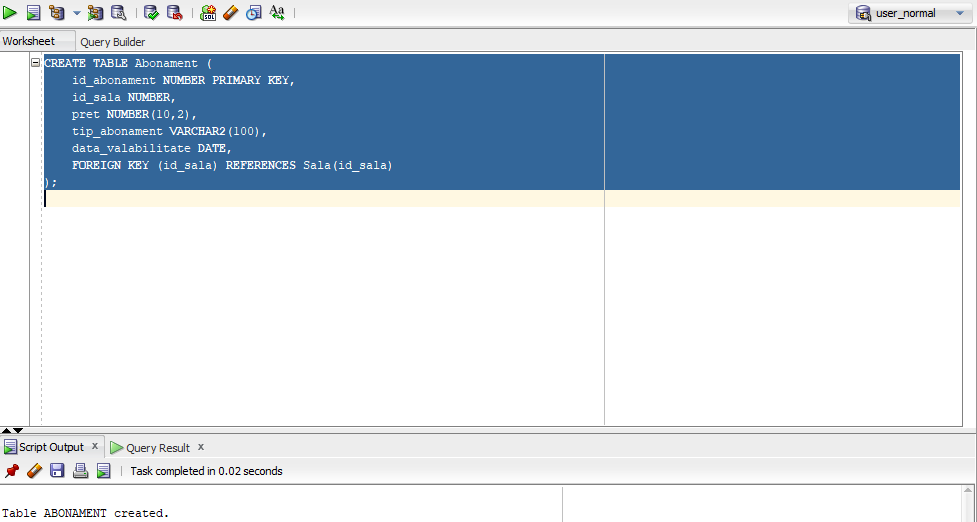


## 

## --4.Pentru tabelul ABONAMENT

| CREATE TABLE Abonament (  id\_abonament NUMBER PRIMARY KEY,  id\_sala NUMBER,  pret NUMBER(10,2),  tip\_abonament VARCHAR2(100),  data\_valabilitate DATE,  FOREIGN KEY (id\_sala) REFERENCES Sala(id\_sala)  ); |
| --- |

Screenshot:

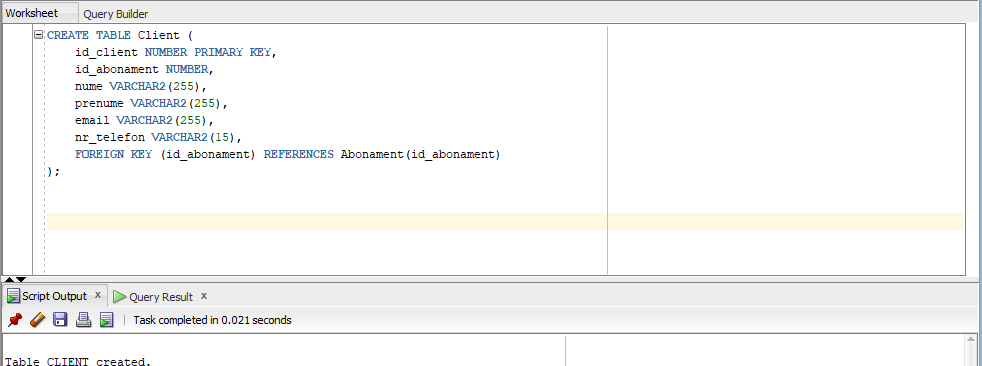


## 

## --5.Pentru tabelul CLIENT

| CREATE TABLE Client (  id\_client NUMBER PRIMARY KEY,  id\_abonament NUMBER,  nume VARCHAR2(255),  prenume VARCHAR2(255),  email VARCHAR2(255),  nr\_telefon VARCHAR2(15),  FOREIGN KEY (id\_abonament) REFERENCES Abonament(id\_abonament)  ); |
| --- |

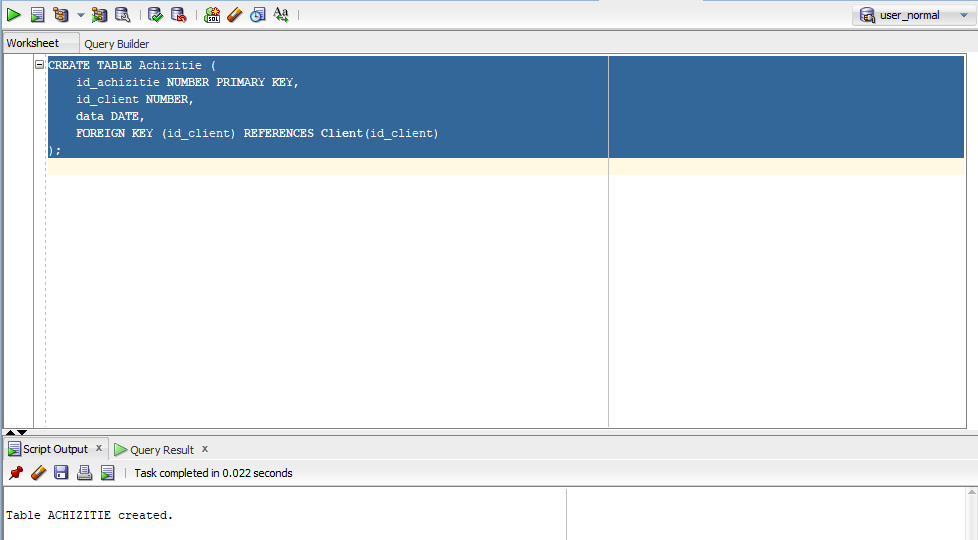
Screenshot:



## --6.Pentru tabelul ACHIZITIE

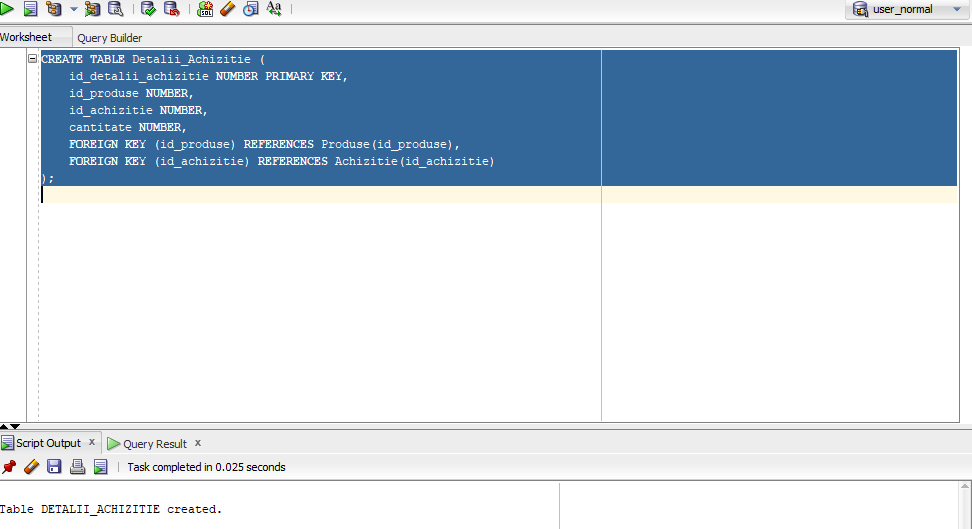
| CREATE TABLE Achizitie (  id\_achizitie NUMBER PRIMARY KEY,  id\_client NUMBER,  data DATE,  FOREIGN KEY (id\_client) REFERENCES Client(id\_client)  ); |
| --- |

Screenshot:



## --7.Pentru tabelul DETALII\_ACHIZITIE

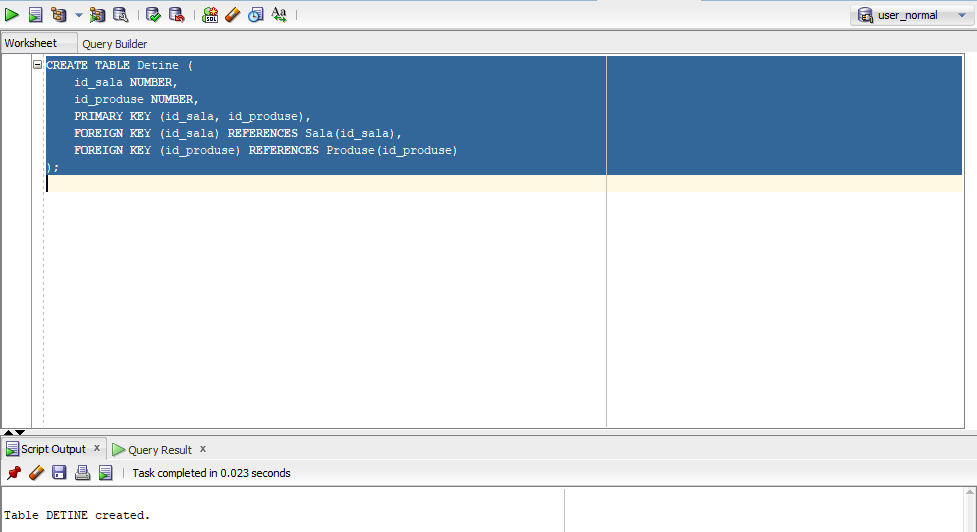
| CREATE TABLE Detalii\_Achizitie (  id\_detalii\_achizitie NUMBER PRIMARY KEY,  id\_produse NUMBER,  id\_achizitie NUMBER,  cantitate NUMBER,  FOREIGN KEY (id\_produse) REFERENCES Produse(id\_produse),  FOREIGN KEY (id\_achizitie) REFERENCES Achizitie(id\_achizitie)  ); |
| --- |

Screenshot:  


## --8.Pentru tabelul DETINE

| CREATE TABLE Detine (  id\_sala NUMBER,  id\_produse NUMBER,  PRIMARY KEY (id\_sala, id\_produse),  FOREIGN KEY (id\_sala) REFERENCES Sala(id\_sala),  FOREIGN KEY (id\_produse) REFERENCES Produse(id\_produse)  ); |
| --- |

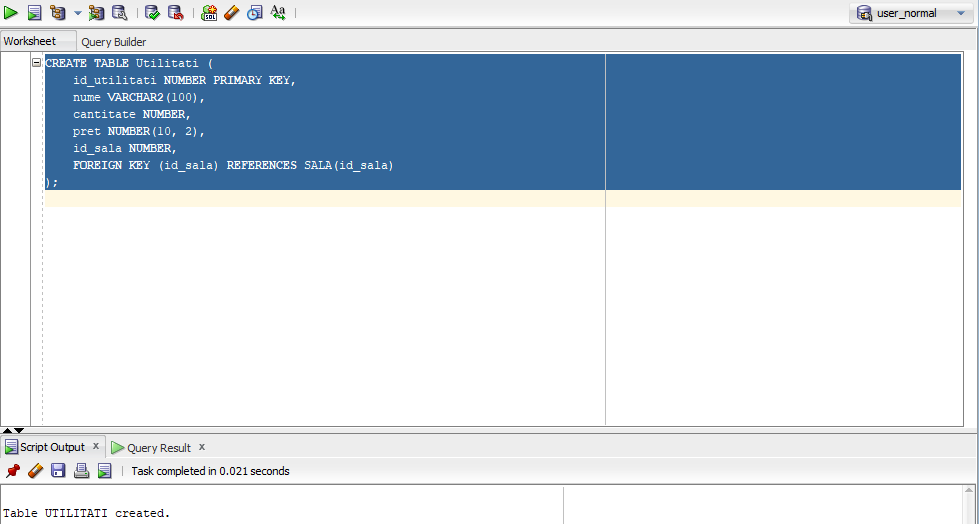
Screenshot:



## --9.Pentru tabelul UTILITATI

| CREATE TABLE Utilitati (  id\_utilitati NUMBER PRIMARY KEY,  nume VARCHAR2(100),  cantitate NUMBER,  pret NUMBER(10, 2),  id\_sala NUMBER,  FOREIGN KEY (id\_sala) REFERENCES SALA(id\_sala)  ); |
| --- |

Screenshot:

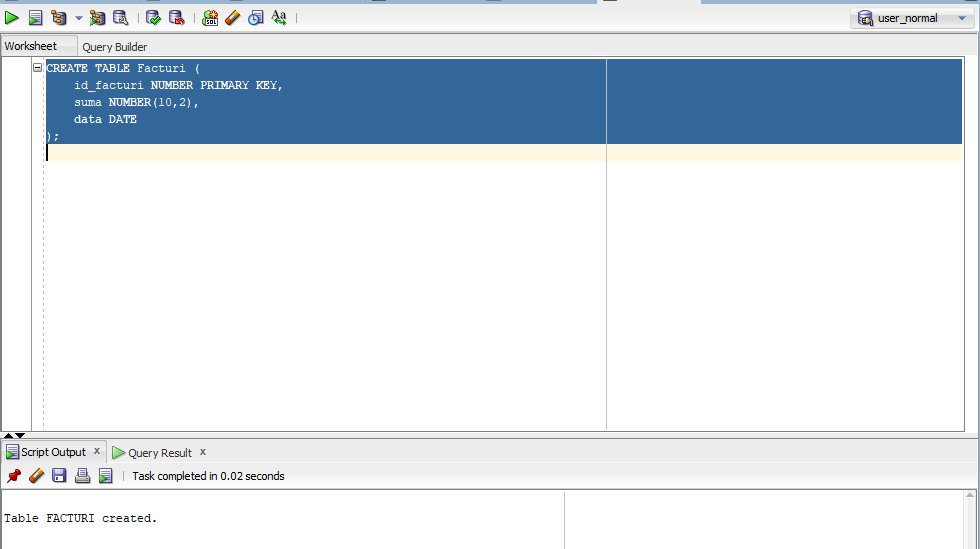


## 

## --10.Pentru tabelul FACTURI

| CREATE TABLE Facturi (  id\_facturi NUMBER PRIMARY KEY,  suma NUMBER(10,2),  data DATE  ); |
| --- |

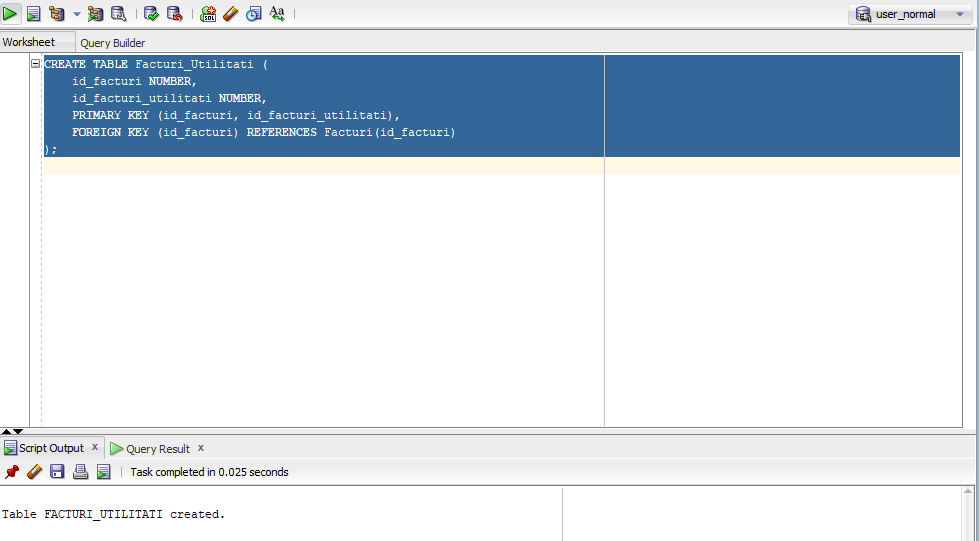
Screenshot:



## --11.Pentru tabelul FACTURI\_UTILITATI

| CREATE TABLE Facturi\_Utilitati (  id\_facturi NUMBER,  id\_facturi\_utilitati NUMBER,  PRIMARY KEY (id\_facturi, id\_facturi\_utilitati),  FOREIGN KEY (id\_facturi) REFERENCES Facturi(id\_facturi)  ); |
| --- |

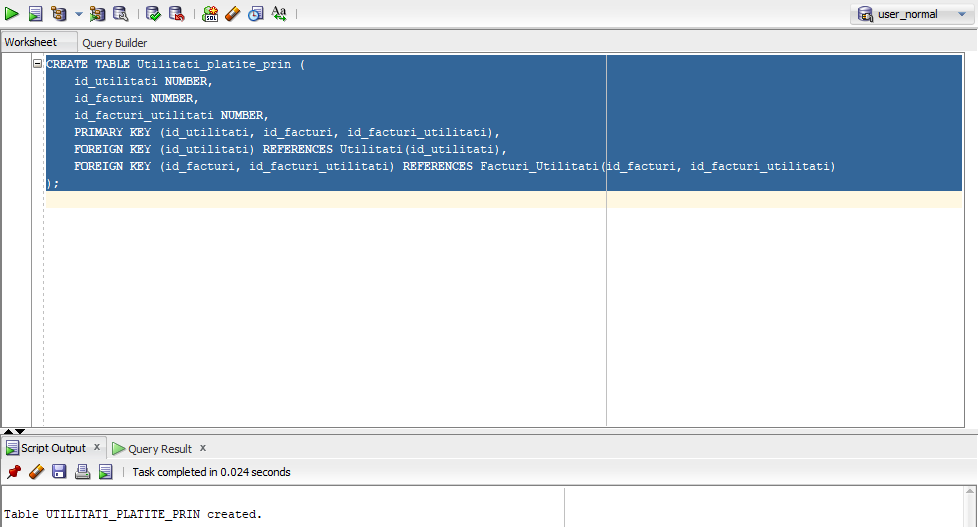
Screenshot:



## --12.Pentru tabelul UTILITATI\_PLATITE\_PRIN

| CREATE TABLE Utilitati\_platite\_prin (  id\_utilitati NUMBER,  id\_facturi NUMBER,  id\_facturi\_utilitati NUMBER,  PRIMARY KEY (id\_utilitati, id\_facturi, id\_facturi\_utilitati),  FOREIGN KEY (id\_utilitati) REFERENCES Utilitati(id\_utilitati),  FOREIGN KEY (id\_facturi, id\_facturi\_utilitati) REFERENCES Facturi\_Utilitati(id\_facturi, id\_facturi\_utilitati)  ); |
| --- |

Screenshot:

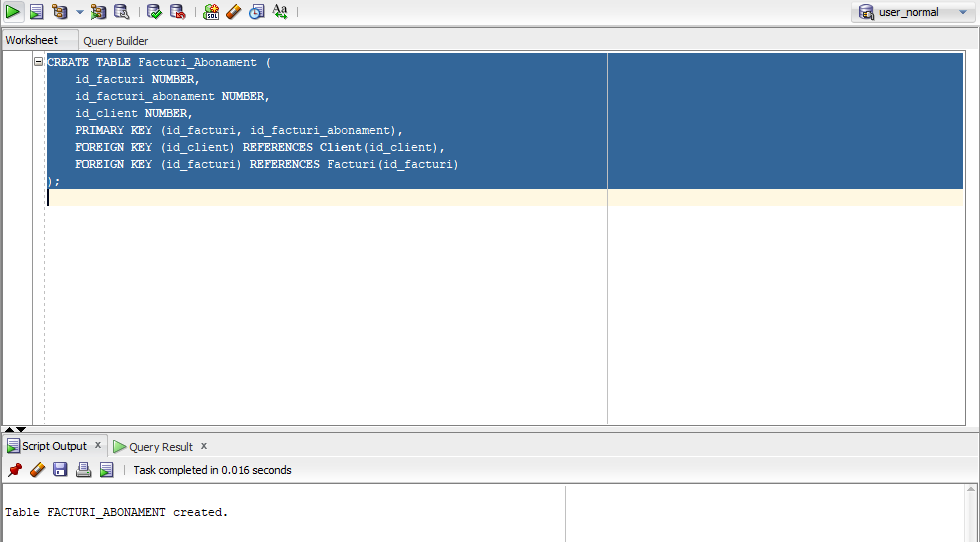


## 

## –13.Pentru tabelul FACTURI\_ABONAMENT

| CREATE TABLE Facturi\_Abonament (  id\_facturi NUMBER,  id\_facturi\_abonament NUMBER,  id\_client NUMBER,  PRIMARY KEY (id\_facturi, id\_facturi\_abonament),  FOREIGN KEY (id\_client) REFERENCES Client(id\_client),  FOREIGN KEY (id\_facturi) REFERENCES Facturi(id\_facturi)  ); |
| --- |

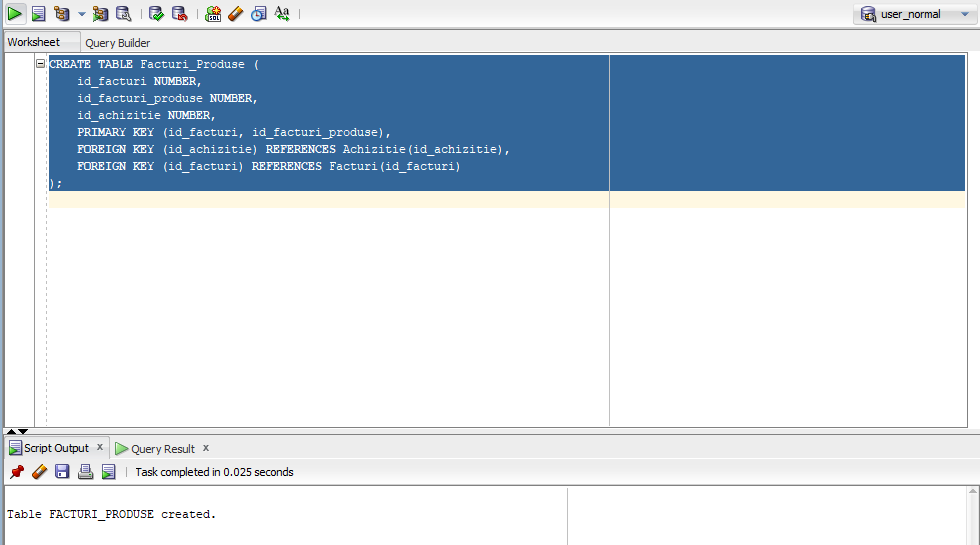
Screenshot:



## --14.Pentru tabelul FACTURI\_PRODUSE

| CREATE TABLE Facturi\_Produse (  id\_facturi NUMBER,  id\_facturi\_produse NUMBER,  id\_achizitie NUMBER,  PRIMARY KEY (id\_facturi, id\_facturi\_produse),  FOREIGN KEY (id\_achizitie) REFERENCES Achizitie(id\_achizitie),  FOREIGN KEY (id\_facturi) REFERENCES Facturi(id\_facturi)  ); |
| --- |

Screenshot:



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

SALA:

* id\_sala (NUMBER, PK): identificatorul unic al sălii
* nr\_clienti (NUMBER): numărul de clienți prezenți în sală
* venituri (NUMBER(10,2)): veniturile generate de sală
* nr\_produse (NUMBER): numărul de produse vândute în sală
* nume\_manager (VARCHAR2(255)): numele managerului sălii
* deschis (VARCHAR2(5)): indicator pentru starea de deschidere a sălii

INSERT INTO Sala (id\_sala, nr\_clienti, venituri, nr\_produse, nume\_manager, deschis)

VALUES (1, 50, 1500.75, 20, 'Ion Popescu', 'Da');

INSERT INTO Sala (id\_sala, nr\_clienti, venituri, nr\_produse, nume\_manager, deschis)

VALUES (2, 30, 1200.50, 15, 'Maria Ionescu', 'Nu');

INSERT INTO Sala (id\_sala, nr\_clienti, venituri, nr\_produse, nume\_manager, deschis)

VALUES (3, 40, 2000.00, 25, 'George Marinescu', 'Da');

INSERT INTO Sala (id\_sala, nr\_clienti, venituri, nr\_produse, nume\_manager, deschis)

VALUES (4, 20, 800.25, 10, 'Elena Popa', 'Nu');

INSERT INTO Sala (id\_sala, nr\_clienti, venituri, nr\_produse, nume\_manager, deschis)

VALUES (5, 60, 2500.90, 30, 'Alexandru Stoica', 'Da');

INSERT INTO Sala (id\_sala, nr\_clienti, venituri, nr\_produse, nume\_manager, deschis)

VALUES (6, 45, 1800.60, 18, 'Ana Radu', 'Nu');

INSERT INTO Sala (id\_sala, nr\_clienti, venituri, nr\_produse, nume\_manager, deschis)

VALUES (7, 55, 2200.35, 22, 'Andrei Dima', 'Da');

INSERT INTO Sala (id\_sala, nr\_clienti, venituri, nr\_produse, nume\_manager, deschis)

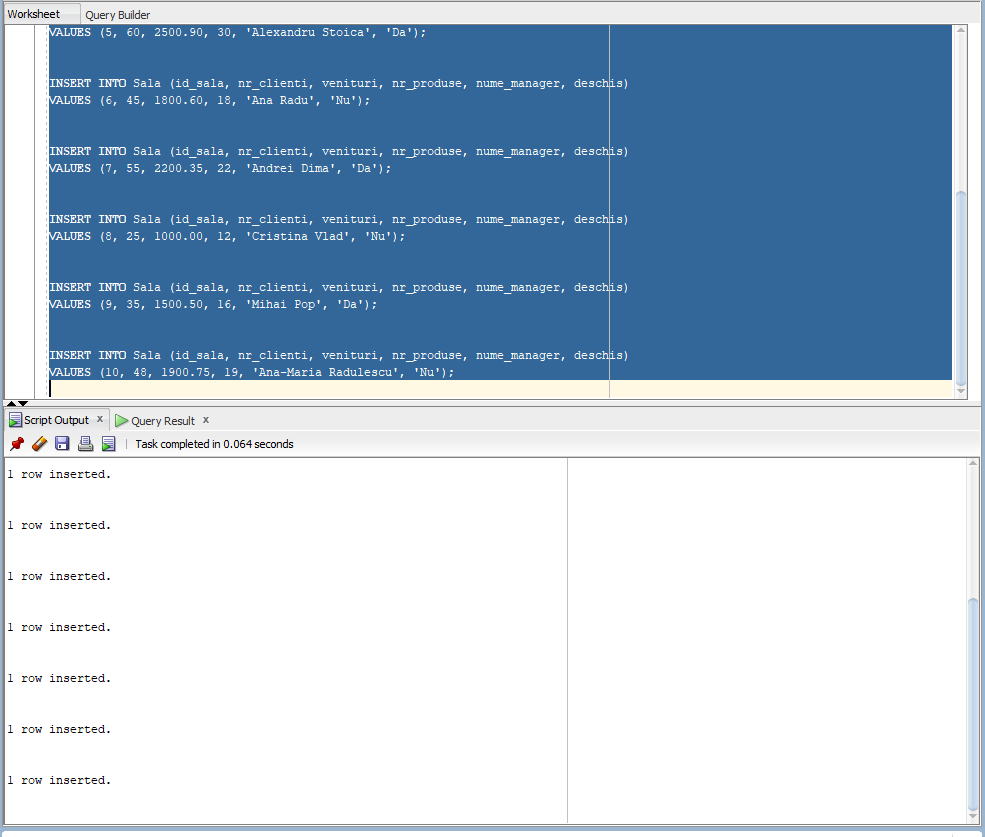
VALUES (8, 25, 1000.00, 12, 'Cristina Vlad', 'Nu');

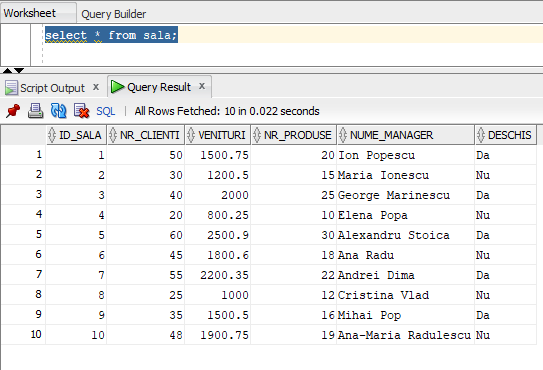
INSERT INTO Sala (id\_sala, nr\_clienti, venituri, nr\_produse, nume\_manager, deschis)

VALUES (9, 35, 1500.50, 16, 'Mihai Pop', 'Da');

INSERT INTO Sala (id\_sala, nr\_clienti, venituri, nr\_produse, nume\_manager, deschis)

VALUES (10, 48, 1900.75, 19, 'Ana-Maria Radulescu', 'Nu');





PRODUSE:

* id\_produse (NUMBER, PK): identificatorul unic al produsului
* nume (VARCHAR2(100)): numele produsului
* cantitate (NUMBER): cantitatea disponibilă a produsului
* pret (NUMBER(10,2)): prețul unitar al produsului

INSERT INTO Produse (id\_produse, nume, cantitate, pret) VALUES (1, 'Proteina', 50, 69.99);

INSERT INTO Produse (id\_produse, nume, cantitate, pret) VALUES (2, 'Bara proteica', 40, 2.99);

INSERT INTO Produse (id\_produse, nume, cantitate, pret) VALUES (3, 'Energy drink', 30, 3.49);

INSERT INTO Produse (id\_produse, nume, cantitate, pret) VALUES (4, 'Multivitamine', 25, 19.99);

INSERT INTO Produse (id\_produse, nume, cantitate, pret) VALUES (5, 'Mix seminte', 20, 7.99);

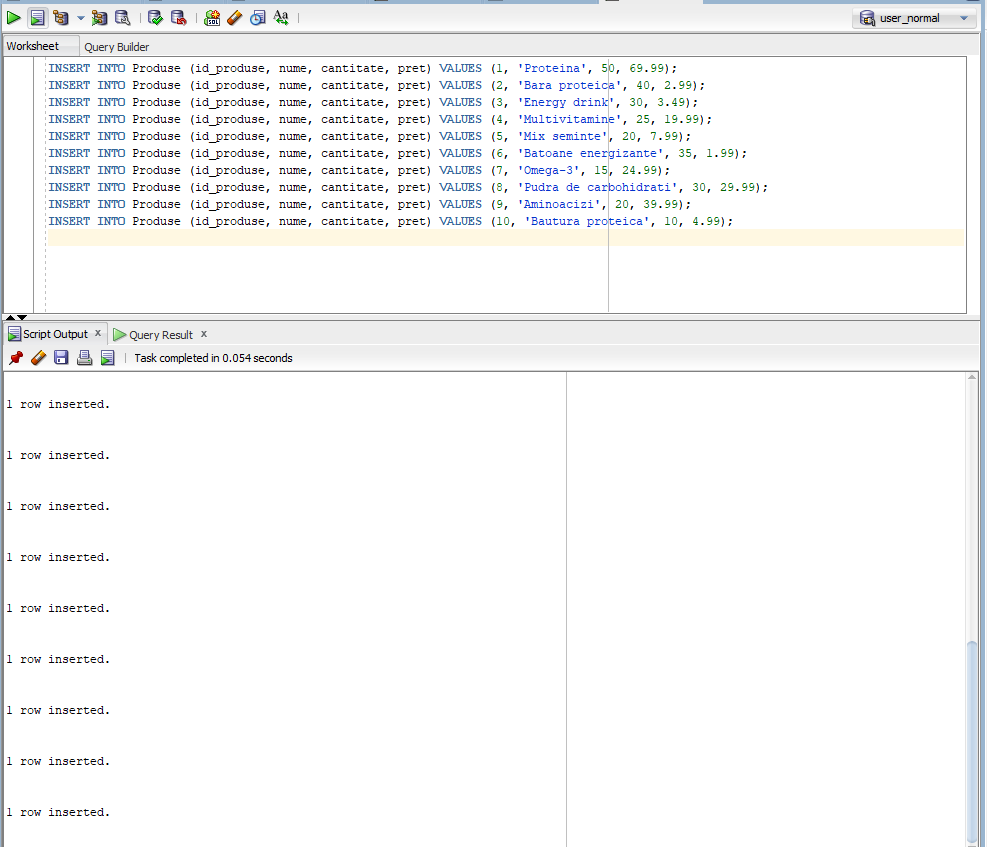
INSERT INTO Produse (id\_produse, nume, cantitate, pret) VALUES (6, 'Batoane energizante', 35, 1.99);

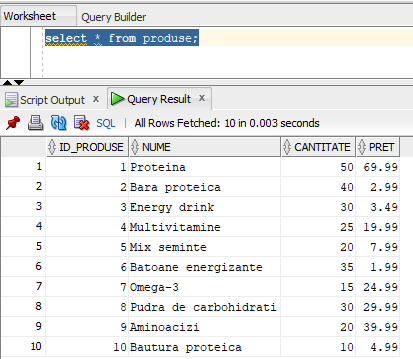
INSERT INTO Produse (id\_produse, nume, cantitate, pret) VALUES (7, 'Omega-3', 15, 24.99);

INSERT INTO Produse (id\_produse, nume, cantitate, pret) VALUES (8, 'Pudra de carbohidrati', 30, 29.99);

INSERT INTO Produse (id\_produse, nume, cantitate, pret) VALUES (9, 'Aminoacizi', 20, 39.99);

INSERT INTO Produse (id\_produse, nume, cantitate, pret) VALUES (10, 'Bautura proteica', 10, 4.99);





APARATE:

* id\_aparate (NUMBER, PK): identificatorul unic al aparatului
* id\_sala (NUMBER, FK): cheia externă către tabela Sala
* nume (VARCHAR2(100)): numele aparatului
* masa\_maxima (NUMBER): masa maximă suportată de aparat
* disponibilitate (VARCHAR2(5)): disponibilitatea aparatului (e.g., 'DA' sau 'NU')
* grupa\_musculara\_lucrata (VARCHAR2(255)): grupa musculară lucrată de aparat

INSERT INTO Aparate (id\_aparat, id\_sala, nume, masa\_maxima, disponibilitate, grupa\_musculara\_lucrata)

VALUES (1, 1, 'Banca abdomene', 150, 'DA', 'Abdomen');

INSERT INTO Aparate (id\_aparat, id\_sala, nume, masa\_maxima, disponibilitate, grupa\_musculara\_lucrata)

VALUES (2, 1, 'Bicicleta', 200, 'DA', 'Cardio');

INSERT INTO Aparate (id\_aparat, id\_sala, nume, masa\_maxima, disponibilitate, grupa\_musculara\_lucrata)

VALUES (3, 2, 'Ghemuit cu gantere', 100, 'NU', 'Biceps');

INSERT INTO Aparate (id\_aparat, id\_sala, nume, masa\_maxima, disponibilitate, grupa\_musculara\_lucrata)

VALUES (4, 2, 'Scripete spate', 180, 'DA', 'Spate');

INSERT INTO Aparate (id\_aparat, id\_sala, nume, masa\_maxima, disponibilitate, grupa\_musculara\_lucrata)

VALUES (5, 3, 'Bench press', 220, 'DA', 'Piept');

INSERT INTO Aparate (id\_aparat, id\_sala, nume, masa\_maxima, disponibilitate, grupa\_musculara\_lucrata)

VALUES (6, 3, 'Leg press', 300, 'DA', 'Picioare');

INSERT INTO Aparate (id\_aparat, id\_sala, nume, masa\_maxima, disponibilitate, grupa\_musculara\_lucrata)

VALUES (7, 4, 'Tractiuni la bara', 180, 'NU', 'Spate');

INSERT INTO Aparate (id\_aparat, id\_sala, nume, masa\_maxima, disponibilitate, grupa\_musculara\_lucrata)

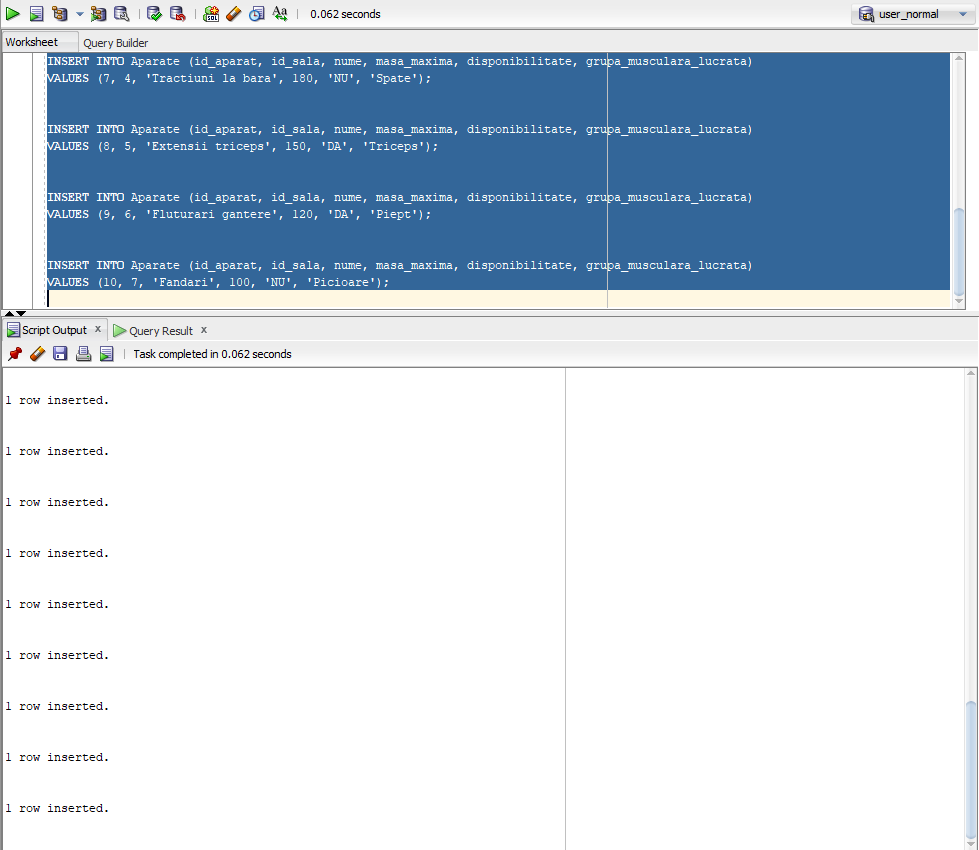
VALUES (8, 5, 'Extensii triceps', 150, 'DA', 'Triceps');

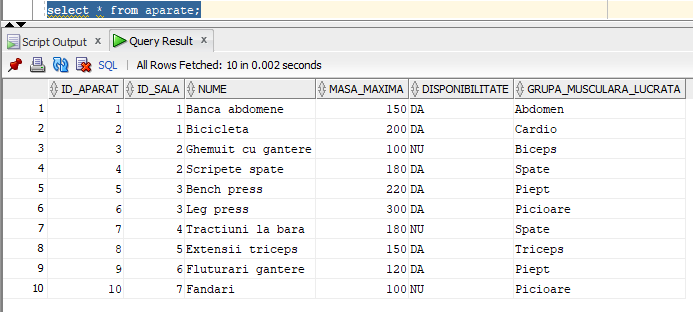
INSERT INTO Aparate (id\_aparat, id\_sala, nume, masa\_maxima, disponibilitate, grupa\_musculara\_lucrata)

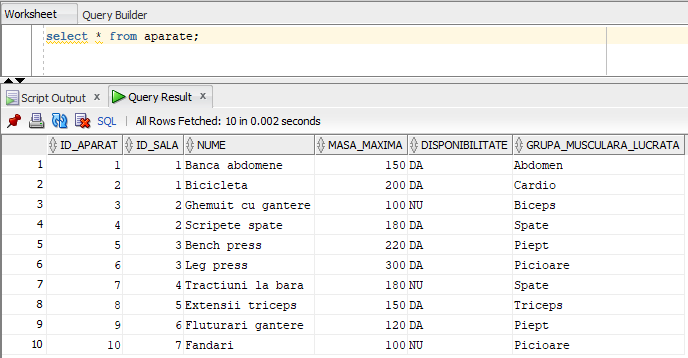
VALUES (9, 6, 'Fluturari gantere', 120, 'DA', 'Piept');

INSERT INTO Aparate (id\_aparat, id\_sala, nume, masa\_maxima, disponibilitate, grupa\_musculara\_lucrata)

VALUES (10, 7, 'Fandari', 100, 'NU', 'Picioare');







ABONAMENT:

* id\_abonament (NUMBER, PK): identificatorul unic al abonamentului
* id\_sala (NUMBER, FK): cheia externă către tabela Sala
* pret (NUMBER(10,2)): prețul abonamentului
* tip\_abonament (VARCHAR2(100)): tipul abonamentului
* data\_valabilitate (DATE): data de expirare a abonamentului

INSERT INTO Abonament (id\_abonament, id\_sala, pret, tip\_abonament, data\_valabilitate) VALUES (1, 1, 150.00, 'Abonament Lunar', TO\_DATE('2024-12-31', 'YYYY-MM-DD'));

INSERT INTO Abonament (id\_abonament, id\_sala, pret, tip\_abonament, data\_valabilitate) VALUES (2, 1, 300.00, 'Abonament Trimestrial', TO\_DATE('2025-03-31', 'YYYY-MM-DD'));

INSERT INTO Abonament (id\_abonament, id\_sala, pret, tip\_abonament, data\_valabilitate) VALUES (3, 2, 200.00, 'Abonament Lunar', TO\_DATE('2024-12-31', 'YYYY-MM-DD'));

INSERT INTO Abonament (id\_abonament, id\_sala, pret, tip\_abonament, data\_valabilitate) VALUES (4, 2, 500.00, 'Abonament Anual', TO\_DATE('2025-12-31', 'YYYY-MM-DD'));

INSERT INTO Abonament (id\_abonament, id\_sala, pret, tip\_abonament, data\_valabilitate) VALUES (5, 3, 180.00, 'Abonament Lunar', TO\_DATE('2024-12-31', 'YYYY-MM-DD'));

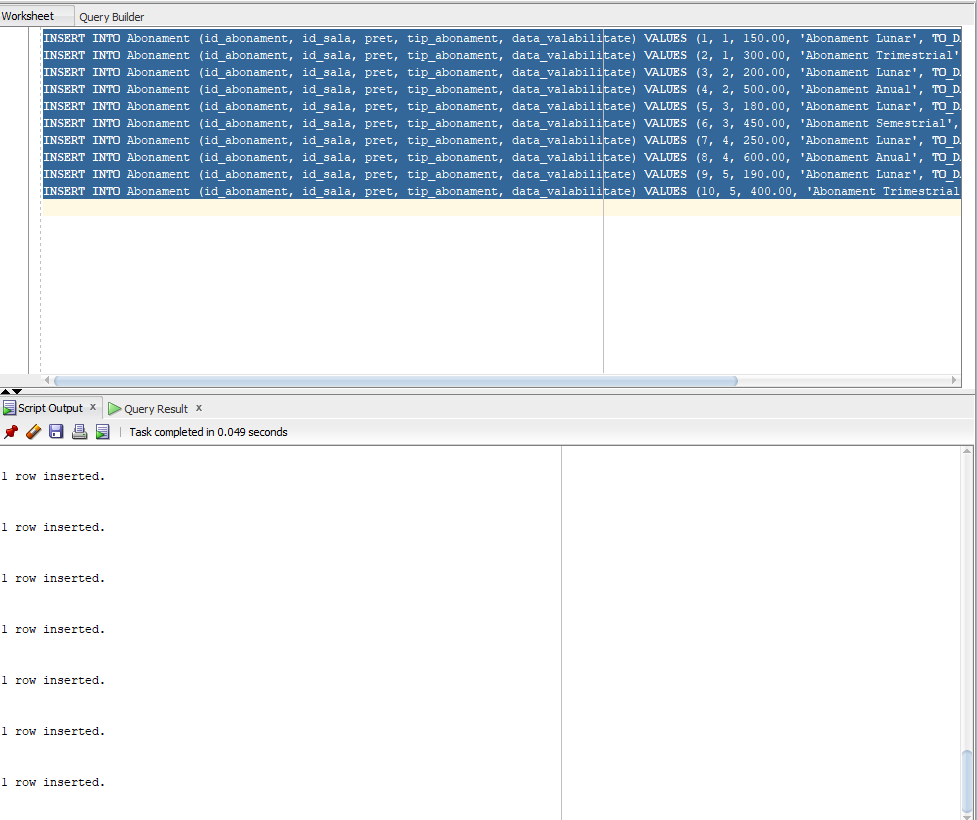
INSERT INTO Abonament (id\_abonament, id\_sala, pret, tip\_abonament, data\_valabilitate) VALUES (6, 3, 450.00, 'Abonament Semestrial', TO\_DATE('2025-06-30', 'YYYY-MM-DD'));

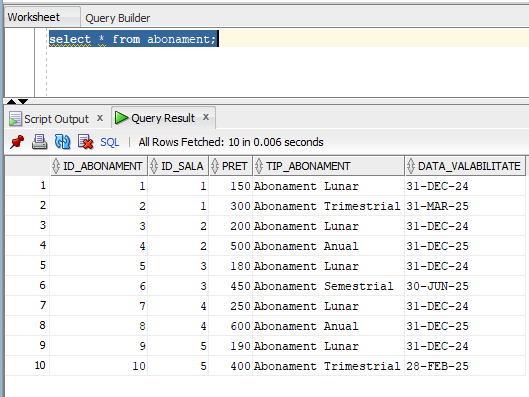
INSERT INTO Abonament (id\_abonament, id\_sala, pret, tip\_abonament, data\_valabilitate) VALUES (7, 4, 250.00, 'Abonament Lunar', TO\_DATE('2024-12-31', 'YYYY-MM-DD'));

INSERT INTO Abonament (id\_abonament, id\_sala, pret, tip\_abonament, data\_valabilitate) VALUES (8, 4, 600.00, 'Abonament Anual', TO\_DATE('2025-12-31', 'YYYY-MM-DD'));

INSERT INTO Abonament (id\_abonament, id\_sala, pret, tip\_abonament, data\_valabilitate) VALUES (9, 5, 190.00, 'Abonament Lunar', TO\_DATE('2024-12-31', 'YYYY-MM-DD'));

INSERT INTO Abonament (id\_abonament, id\_sala, pret, tip\_abonament, data\_valabilitate) VALUES (10, 5, 400.00, 'Abonament Trimestrial', TO\_DATE('2025-02-28', 'YYYY-MM-DD'));





CLIENT:

* id\_client (NUMBER, PK): identificatorul unic al clientului
* id\_abonament (NUMBER, FK): cheia externă către tabela Abonament
* nume (VARCHAR2(255)): numele clientului
* prenume (VARCHAR2(255)): prenumele clientului
* email (VARCHAR2(255)): adresa de email a clientului
* nr\_telefon (VARCHAR2(15)): numărul de telefon al clientului

INSERT INTO Client (id\_client, id\_abonament, nume, prenume, email, nr\_telefon)

VALUES (1, 1, 'Popescu', 'Ion', 'popescu.ion@gmail.com', '0721123456');

INSERT INTO Client (id\_client, id\_abonament, nume, prenume, email, nr\_telefon)

VALUES (2, 2, 'Ionescu', 'Maria', 'ionescu.maria@gmail.com', '0732123456');

INSERT INTO Client (id\_client, id\_abonament, nume, prenume, email, nr\_telefon)

VALUES (3, 3, 'Pop', 'Ana', 'pop.ana@gmail.com', '0743123456');

INSERT INTO Client (id\_client, id\_abonament, nume, prenume, email, nr\_telefon)

VALUES (4, 4, 'Georgescu', 'Andrei', 'georgescu.andrei@gmail.com', '0754123456');

INSERT INTO Client (id\_client, id\_abonament, nume, prenume, email, nr\_telefon)

VALUES (5, 5, 'Popa', 'Elena', 'popa.elena@gmail.com', '0765123456');

INSERT INTO Client (id\_client, id\_abonament, nume, prenume, email, nr\_telefon)

VALUES (6, 6, 'Dragomir', 'Mihai', 'dragomir.mihai@gmail.com', '0776123456');

INSERT INTO Client (id\_client, id\_abonament, nume, prenume, email, nr\_telefon)

VALUES (7, 7, 'Stancu', 'Ioana', 'stancu.ioana@gmail.com', '0787123456');

INSERT INTO Client (id\_client, id\_abonament, nume, prenume, email, nr\_telefon)

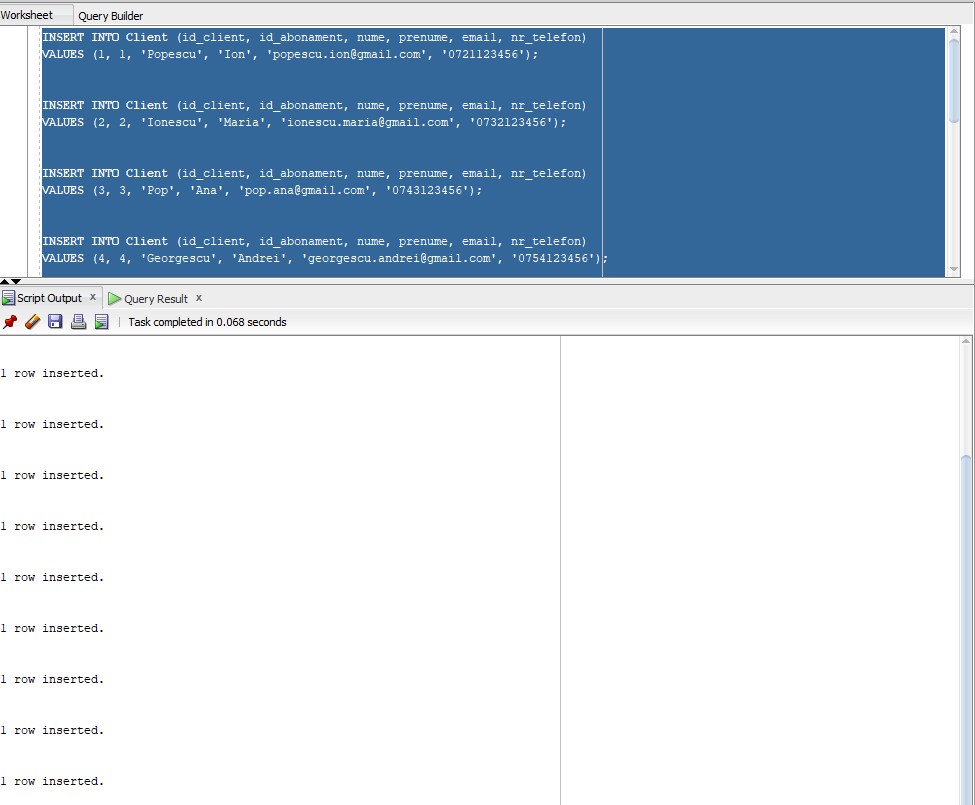
VALUES (8, 8, 'Dumitrescu', 'Alex', 'dumitrescu.alex@gmail.com', '0798123456');

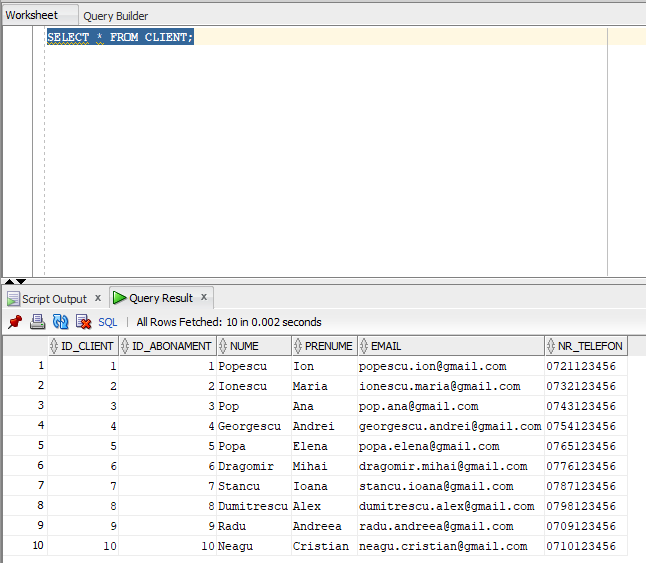
INSERT INTO Client (id\_client, id\_abonament, nume, prenume, email, nr\_telefon)

VALUES (9, 9, 'Radu', 'Andreea', 'radu.andreea@gmail.com', '0709123456');

INSERT INTO Client (id\_client, id\_abonament, nume, prenume, email, nr\_telefon)

VALUES (10, 10, 'Neagu', 'Cristian', 'neagu.cristian@gmail.com', '0710123456');





ACHIZITIE:

* id\_achizitie (NUMBER, PK): identificatorul unic al achiziției
* id\_client (NUMBER, FK): cheia externă către tabela Client
* data (DATE): data achiziției

INSERT INTO Achizitie (id\_achizitie, id\_client, data) VALUES (1, 1, TO\_DATE('2024-12-04', 'YYYY-MM-DD'));

INSERT INTO Achizitie (id\_achizitie, id\_client, data) VALUES (2, 2, TO\_DATE('2024-12-08', 'YYYY-MM-DD'));

INSERT INTO Achizitie (id\_achizitie, id\_client, data) VALUES (3, 3, TO\_DATE('2024-12-06', 'YYYY-MM-DD'));

INSERT INTO Achizitie (id\_achizitie, id\_client, data) VALUES (4, 4, TO\_DATE('2024-12-01', 'YYYY-MM-DD'));

INSERT INTO Achizitie (id\_achizitie, id\_client, data) VALUES (5, 5, TO\_DATE('2024-12-09', 'YYYY-MM-DD'));

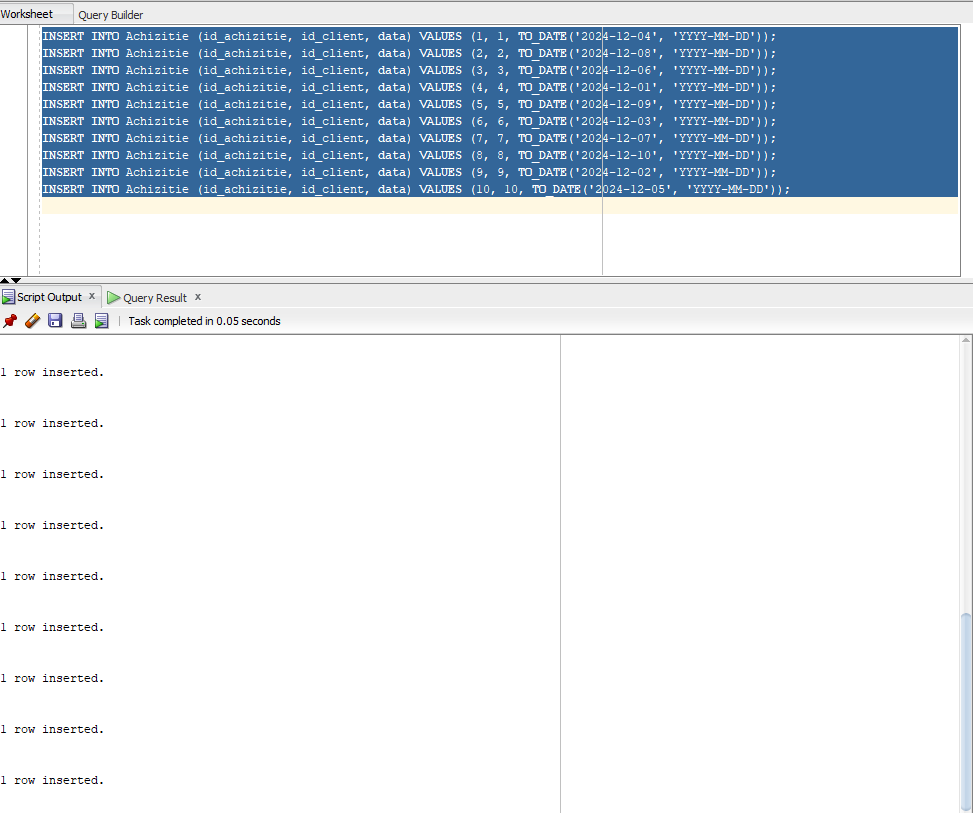
INSERT INTO Achizitie (id\_achizitie, id\_client, data) VALUES (6, 6, TO\_DATE('2024-12-03', 'YYYY-MM-DD'));

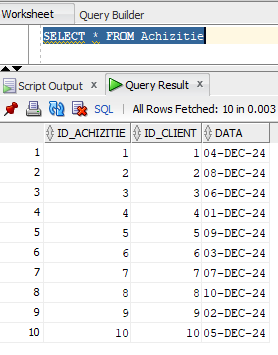
INSERT INTO Achizitie (id\_achizitie, id\_client, data) VALUES (7, 7, TO\_DATE('2024-12-07', 'YYYY-MM-DD'));

INSERT INTO Achizitie (id\_achizitie, id\_client, data) VALUES (8, 8, TO\_DATE('2024-12-10', 'YYYY-MM-DD'));

INSERT INTO Achizitie (id\_achizitie, id\_client, data) VALUES (9, 9, TO\_DATE('2024-12-02', 'YYYY-MM-DD'));

INSERT INTO Achizitie (id\_achizitie, id\_client, data) VALUES (10, 10, TO\_DATE('2024-12-05', 'YYYY-MM-DD'));





DETALII\_ACHIZITIE:

* id\_detalii\_achizitie (NUMBER, PK): identificatorul unic al detaliilor achiziției
* id\_produse (NUMBER, FK): cheia externă către tabela Produse
* id\_achizitie (NUMBER, FK): cheia externă către tabela Achizitie
* cantitate (NUMBER): cantitatea de produse achiziționate

INSERT INTO Detalii\_Achizitie (id\_detalii\_achizitie, id\_produse, id\_achizitie, cantitate) VALUES (1, 1, 1, 5);

INSERT INTO Detalii\_Achizitie (id\_detalii\_achizitie, id\_produse, id\_achizitie, cantitate) VALUES (2, 2, 2, 10);

INSERT INTO Detalii\_Achizitie (id\_detalii\_achizitie, id\_produse, id\_achizitie, cantitate) VALUES (3, 3, 3, 3);

INSERT INTO Detalii\_Achizitie (id\_detalii\_achizitie, id\_produse, id\_achizitie, cantitate) VALUES (4, 4, 4, 8);

INSERT INTO Detalii\_Achizitie (id\_detalii\_achizitie, id\_produse, id\_achizitie, cantitate) VALUES (5, 5, 5, 2);

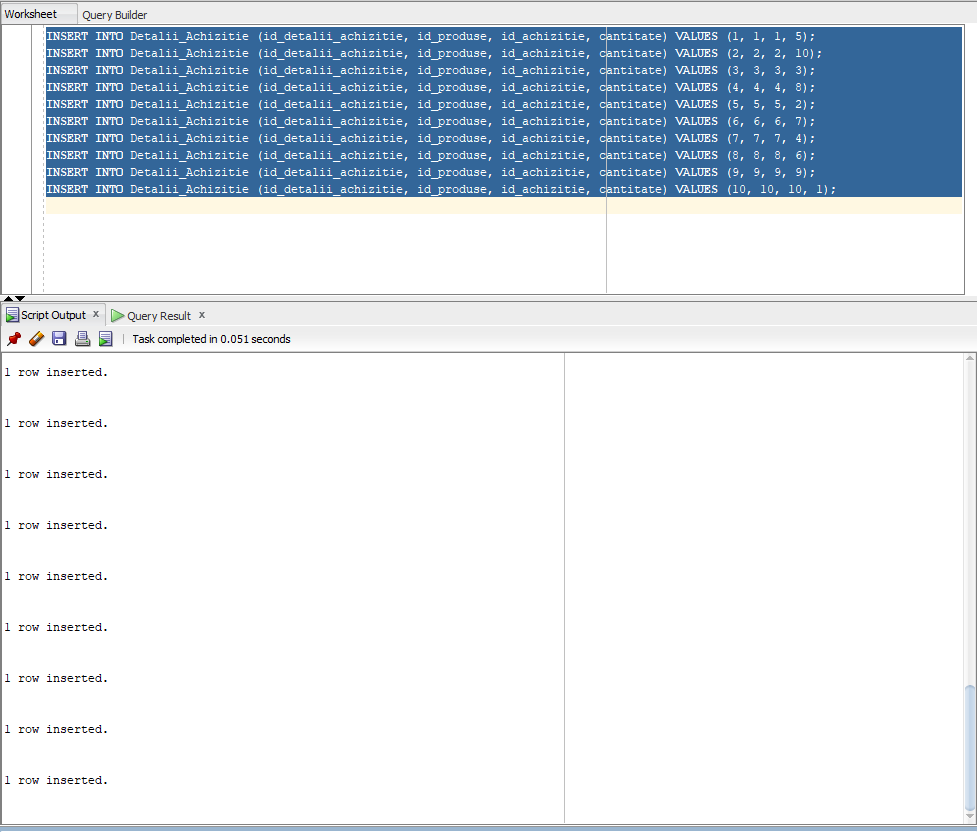
INSERT INTO Detalii\_Achizitie (id\_detalii\_achizitie, id\_produse, id\_achizitie, cantitate) VALUES (6, 6, 6, 7);

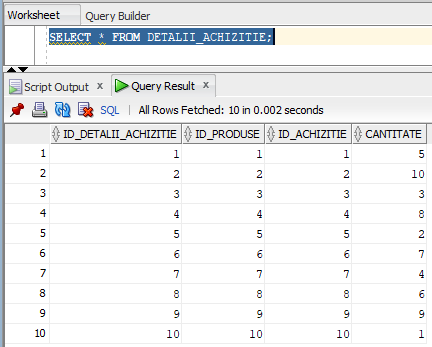
INSERT INTO Detalii\_Achizitie (id\_detalii\_achizitie, id\_produse, id\_achizitie, cantitate) VALUES (7, 7, 7, 4);

INSERT INTO Detalii\_Achizitie (id\_detalii\_achizitie, id\_produse, id\_achizitie, cantitate) VALUES (8, 8, 8, 6);

INSERT INTO Detalii\_Achizitie (id\_detalii\_achizitie, id\_produse, id\_achizitie, cantitate) VALUES (9, 9, 9, 9);

INSERT INTO Detalii\_Achizitie (id\_detalii\_achizitie, id\_produse, id\_achizitie, cantitate) VALUES (10, 10, 10, 1);





DETINE:

* id\_sala (NUMBER, FK, PK): cheia primară și cheia externă către tabela Sala
* id\_produse (NUMBER, FK, PK): cheia primară și cheia externă către tabela Produse

INSERT INTO Detine (id\_sala, id\_produse) VALUES (1, 1);

INSERT INTO Detine (id\_sala, id\_produse) VALUES (1, 2);

INSERT INTO Detine (id\_sala, id\_produse) VALUES (2, 3);

INSERT INTO Detine (id\_sala, id\_produse) VALUES (2, 4);

INSERT INTO Detine (id\_sala, id\_produse) VALUES (3, 5);

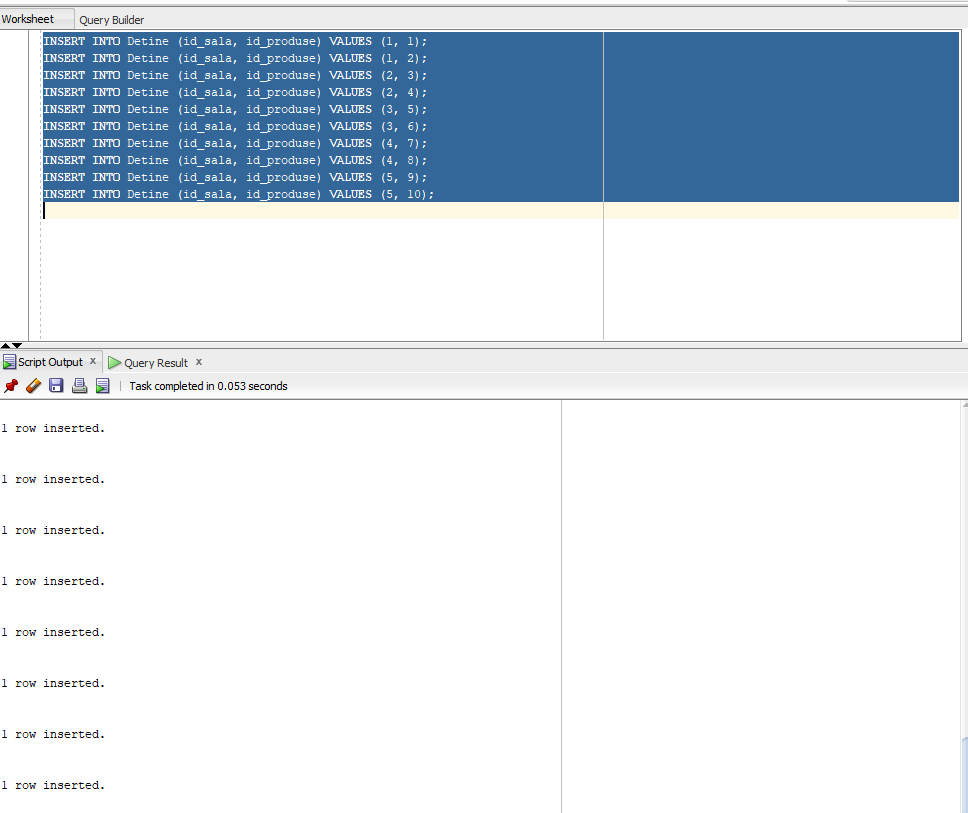
INSERT INTO Detine (id\_sala, id\_produse) VALUES (3, 6);

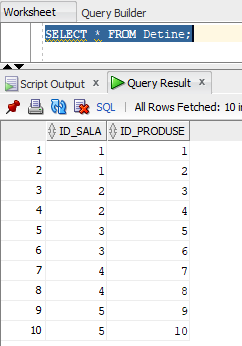
INSERT INTO Detine (id\_sala, id\_produse) VALUES (4, 7);

INSERT INTO Detine (id\_sala, id\_produse) VALUES (4, 8);

INSERT INTO Detine (id\_sala, id\_produse) VALUES (5, 9);

INSERT INTO Detine (id\_sala, id\_produse) VALUES (5, 10);





UTILITATI:

* id\_utilitati(NUMBER, PK): identificatorul unic a utilității.
* id\_sala (NUMBER, FK): cheia externă către tabela Sala
* nume (VARCHAR2(100)): numele utilității
* cantitate (NUMBER): cantitatea disponibilă a utilității
* pret (NUMBER(10,2)): prețul unitar al utilității

INSERT INTO Utilitati (id\_utilitati, id\_sala, nume, cantitate, pret) VALUES (1, 1, 'Gaze naturale', 100, 150.00);

INSERT INTO Utilitati (id\_utilitati, id\_sala, nume, cantitate, pret) VALUES (2, 2, 'Electricitate', 200, 200.00);

INSERT INTO Utilitati (id\_utilitati, id\_sala, nume, cantitate, pret) VALUES (3, 3, 'Apa calda', 150, 100.00);

INSERT INTO Utilitati (id\_utilitati, id\_sala, nume, cantitate, pret) VALUES (4, 4, 'Apa rece', 300, 80.00);

INSERT INTO Utilitati (id\_utilitati, id\_sala, nume, cantitate, pret) VALUES (5, 5, 'Intretinere', 50, 300.00);

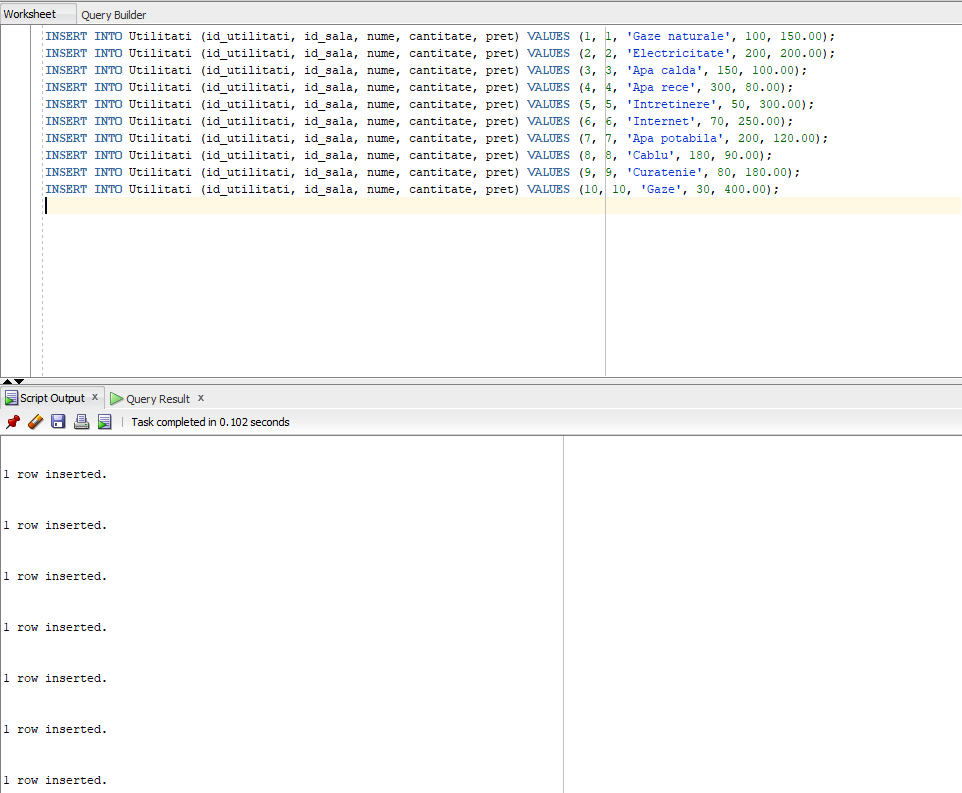
INSERT INTO Utilitati (id\_utilitati, id\_sala, nume, cantitate, pret) VALUES (6, 6, 'Internet', 70, 250.00);

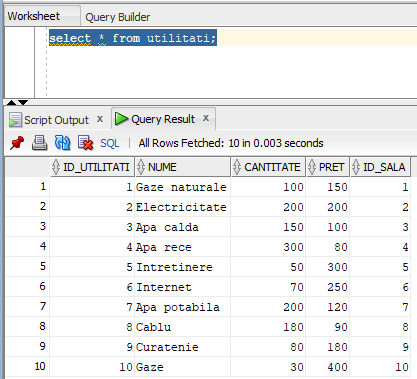
INSERT INTO Utilitati (id\_utilitati, id\_sala, nume, cantitate, pret) VALUES (7, 7, 'Apa potabila', 200, 120.00);

INSERT INTO Utilitati (id\_utilitati, id\_sala, nume, cantitate, pret) VALUES (8, 8, 'Cablu', 180, 90.00);

INSERT INTO Utilitati (id\_utilitati, id\_sala, nume, cantitate, pret) VALUES (9, 9, 'Curatenie', 80, 180.00);

INSERT INTO Utilitati (id\_utilitati, id\_sala, nume, cantitate, pret) VALUES (10, 10, 'Gaze', 30, 400.00);





FACTURI:

* id\_facturi (NUMBER, PK): identificatorul unic al facturii
* id\_sala (NUMBER, FK): cheia externă către tabela Sala
* suma (NUMBER(10,2)): suma facturată
* data (DATE): data facturării

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (1, 500.00, TO\_DATE('2024-12-01', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (2, 750.00, TO\_DATE('2024-12-03', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (3, 600.00, TO\_DATE('2024-12-05', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (4, 900.00, TO\_DATE('2024-12-07', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (5, 550.00, TO\_DATE('2024-12-09', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (6, 400.00, TO\_DATE('2024-12-10', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (7, 700.00, TO\_DATE('2024-12-12', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (8, 650.00, TO\_DATE('2024-12-14', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (9, 850.00, TO\_DATE('2024-12-16', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (10, 500.00, TO\_DATE('2024-12-18', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (11, 450.00, TO\_DATE('2024-12-20', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (12, 800.00, TO\_DATE('2024-12-22', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (13, 670.00, TO\_DATE('2024-12-24', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (14, 950.00, TO\_DATE('2024-12-26', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (15, 600.00, TO\_DATE('2024-12-28', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (16, 420.00, TO\_DATE('2024-12-30', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (17, 730.00, TO\_DATE('2025-01-01', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (18, 640.00, TO\_DATE('2025-01-03', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (19, 920.00, TO\_DATE('2025-01-05', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (20, 580.00, TO\_DATE('2025-01-07', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (21, 470.00, TO\_DATE('2025-01-09', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (22, 760.00, TO\_DATE('2025-01-11', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (23, 630.00, TO\_DATE('2025-01-13', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (24, 880.00, TO\_DATE('2025-01-15', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (25, 590.00, TO\_DATE('2025-01-17', 'YYYY-MM-DD'));

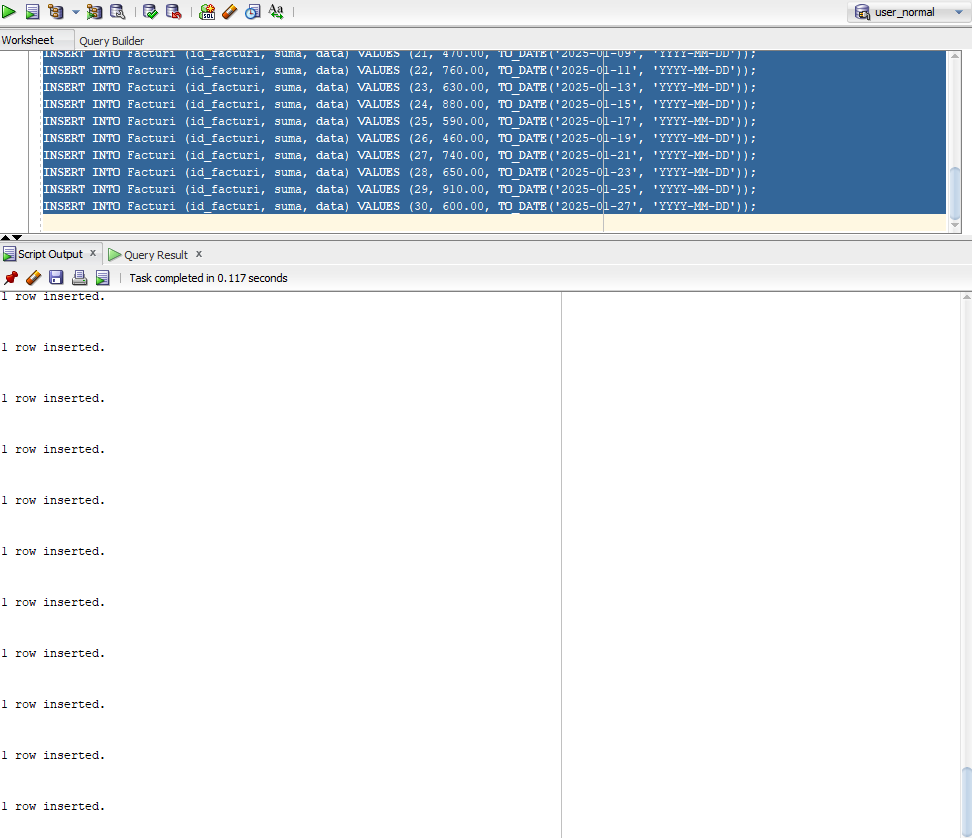
INSERT INTO Facturi (id\_facturi, suma, data) VALUES (26, 460.00, TO\_DATE('2025-01-19', 'YYYY-MM-DD'));

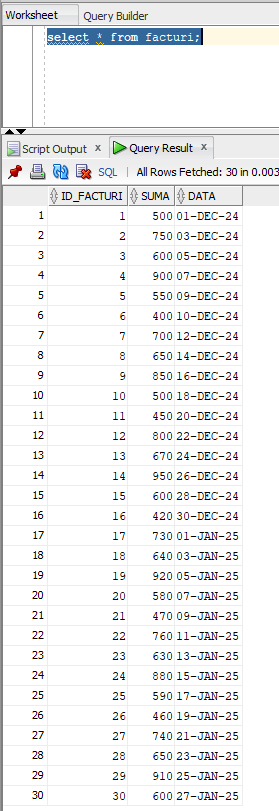
INSERT INTO Facturi (id\_facturi, suma, data) VALUES (27, 740.00, TO\_DATE('2025-01-21', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (28, 650.00, TO\_DATE('2025-01-23', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (29, 910.00, TO\_DATE('2025-01-25', 'YYYY-MM-DD'));

INSERT INTO Facturi (id\_facturi, suma, data) VALUES (30, 600.00, TO\_DATE('2025-01-27', 'YYYY-MM-DD'));





(facturile cu id de la 1 la 10, sunt pentru facturi\_utilitati)

(de la 11 la 20, pentru facturi\_abonament)

(de la 21 la 30, pentru facturi\_produse)

+ o sa creez un trigger pe insert, sa nu ma lase sa inserez 2 facturi (chiar daca sunt din categorii diferite) cu acelasi id\_factura (princpal)

Adica sa nu pot avea factura\_utilitati cu id (1,2) unde 1 este de la id\_facturi și 2 este de la id\_factura\_utilitati; și factura\_abonamente cu id(1,3) unde 1 este id\_facturi și 3 de la id\_ facturi\_abonamente

FACTURI\_UTILITATI:

* id\_facturi\_utilitati (NUMBER, PK): identificatorul unic al facturii pentru utilități
* id\_facturi (NUMBER, FK): cheia externă către tabela Facturi

INSERT INTO Facturi\_Utilitati (id\_facturi, id\_facturi\_utilitati) VALUES (1, 1);

INSERT INTO Facturi\_Utilitati (id\_facturi, id\_facturi\_utilitati) VALUES (2, 2);

INSERT INTO Facturi\_Utilitati (id\_facturi, id\_facturi\_utilitati) VALUES (3, 3);

INSERT INTO Facturi\_Utilitati (id\_facturi, id\_facturi\_utilitati) VALUES (4, 4);

INSERT INTO Facturi\_Utilitati (id\_facturi, id\_facturi\_utilitati) VALUES (5, 5);

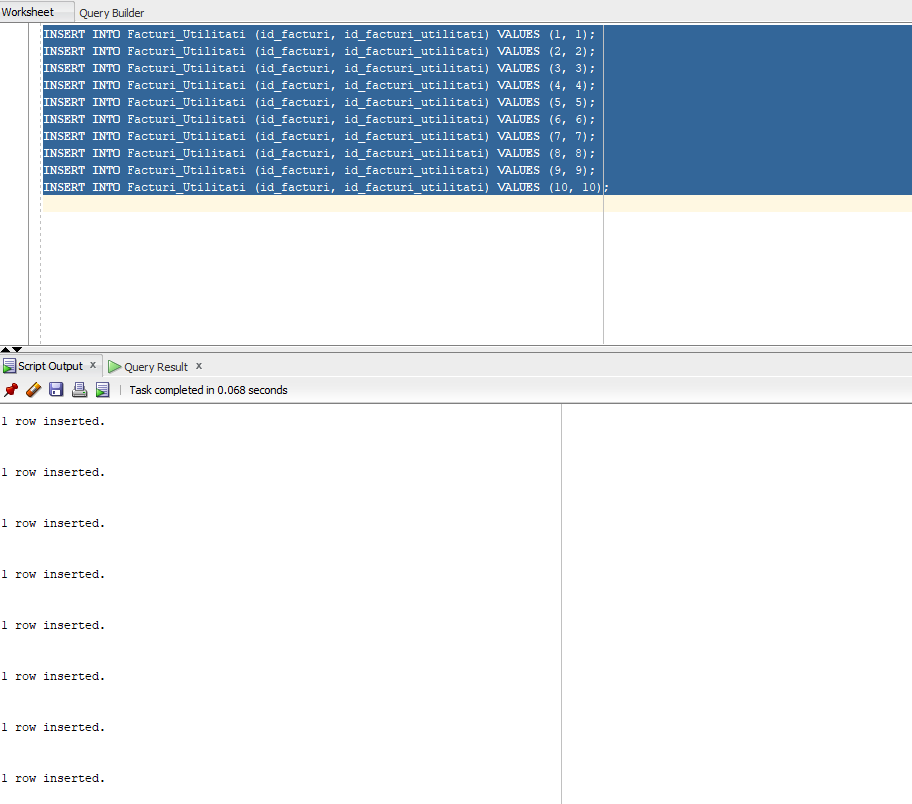
INSERT INTO Facturi\_Utilitati (id\_facturi, id\_facturi\_utilitati) VALUES (6, 6);

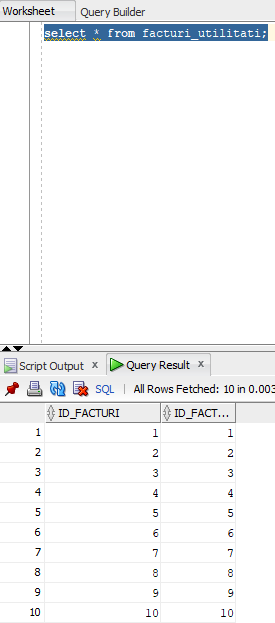
INSERT INTO Facturi\_Utilitati (id\_facturi, id\_facturi\_utilitati) VALUES (7, 7);

INSERT INTO Facturi\_Utilitati (id\_facturi, id\_facturi\_utilitati) VALUES (8, 8);

INSERT INTO Facturi\_Utilitati (id\_facturi, id\_facturi\_utilitati) VALUES (9, 9);

INSERT INTO Facturi\_Utilitati (id\_facturi, id\_facturi\_utilitati) VALUES (10, 10);





UTILITATI\_PLATITE\_PRIN:

* id\_utilitati (NUMBER, FK, PK): cheia primară și cheia externă către tabela Utilitati
* id\_facturi\_utilitati (NUMBER, FK, PK): cheia primară și cheia externă către tabela Facturi\_Utilitati

INSERT INTO Utilitati\_platite\_prin (id\_utilitati, id\_facturi\_utilitati, id\_facturi) VALUES (1, 1, 1);

INSERT INTO Utilitati\_platite\_prin (id\_utilitati, id\_facturi\_utilitati, id\_facturi) VALUES (2, 2, 2);

INSERT INTO Utilitati\_platite\_prin (id\_utilitati, id\_facturi\_utilitati, id\_facturi) VALUES (3, 3, 3);

INSERT INTO Utilitati\_platite\_prin (id\_utilitati, id\_facturi\_utilitati, id\_facturi) VALUES (4, 4, 4);

INSERT INTO Utilitati\_platite\_prin (id\_utilitati, id\_facturi\_utilitati, id\_facturi) VALUES (5, 5, 5);

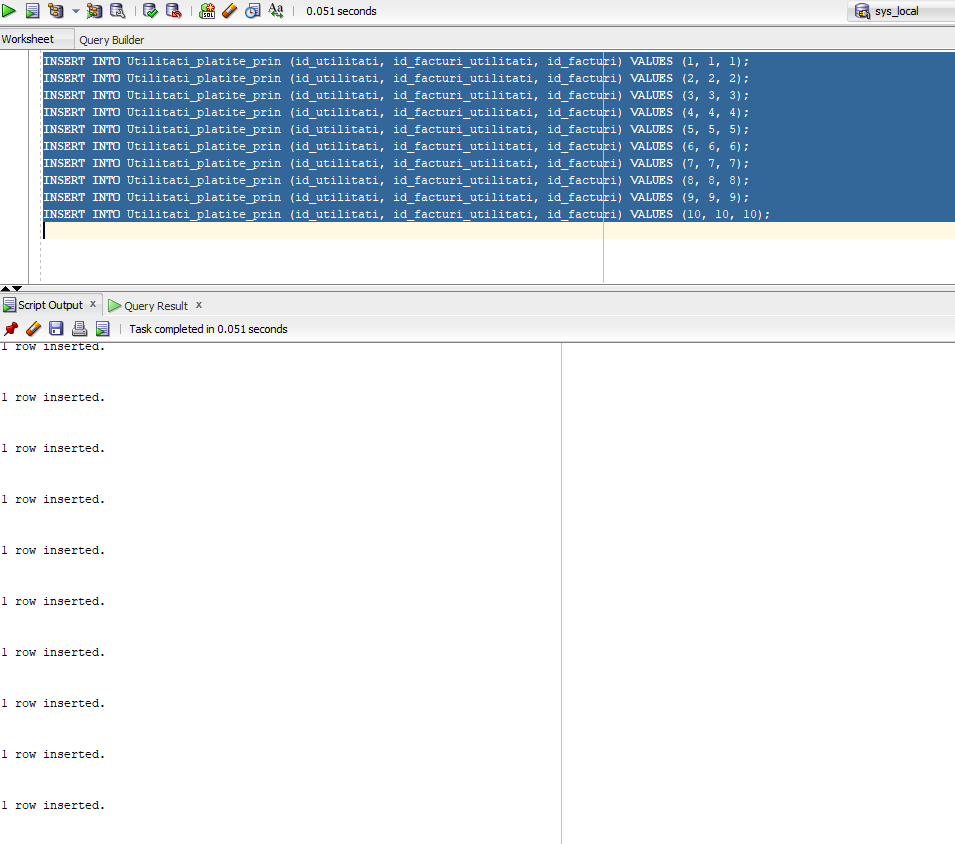
INSERT INTO Utilitati\_platite\_prin (id\_utilitati, id\_facturi\_utilitati, id\_facturi) VALUES (6, 6, 6);

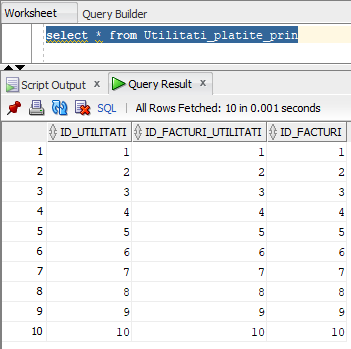
INSERT INTO Utilitati\_platite\_prin (id\_utilitati, id\_facturi\_utilitati, id\_facturi) VALUES (7, 7, 7);

INSERT INTO Utilitati\_platite\_prin (id\_utilitati, id\_facturi\_utilitati, id\_facturi) VALUES (8, 8, 8);

INSERT INTO Utilitati\_platite\_prin (id\_utilitati, id\_facturi\_utilitati, id\_facturi) VALUES (9, 9, 9);

INSERT INTO Utilitati\_platite\_prin (id\_utilitati, id\_facturi\_utilitati, id\_facturi) VALUES (10, 10, 10);





FACTURI\_ABONAMENT:

* id\_facturi\_abonament (NUMBER, PK): identificatorul unic al facturii pentru abonament
* id\_client (NUMBER, FK): cheia externă către tabela Client
* id\_facturi (NUMBER, FK): cheia externă către tabela Facturi

INSERT INTO Facturi\_Abonament (id\_facturi\_abonament, id\_client, id\_facturi) VALUES (1, 1, 11);

INSERT INTO Facturi\_Abonament (id\_facturi\_abonament, id\_client, id\_facturi) VALUES (2, 2, 12);

INSERT INTO Facturi\_Abonament (id\_facturi\_abonament, id\_client, id\_facturi) VALUES (3, 3, 13);

INSERT INTO Facturi\_Abonament (id\_facturi\_abonament, id\_client, id\_facturi) VALUES (4, 4, 14);

INSERT INTO Facturi\_Abonament (id\_facturi\_abonament, id\_client, id\_facturi) VALUES (5, 5, 15);

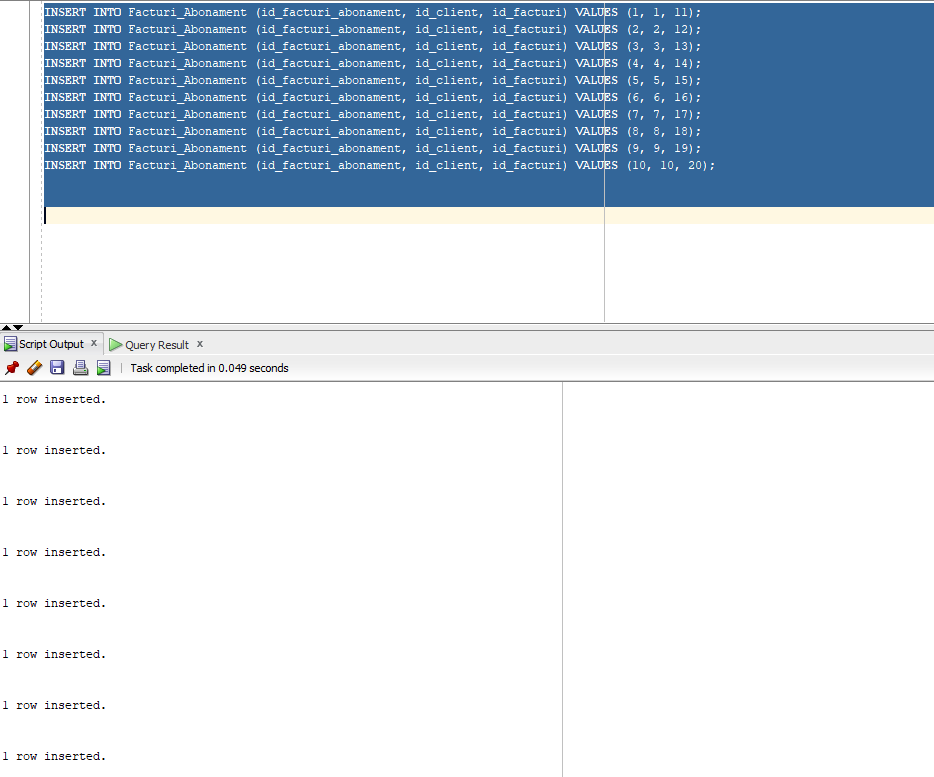
INSERT INTO Facturi\_Abonament (id\_facturi\_abonament, id\_client, id\_facturi) VALUES (6, 6, 16);

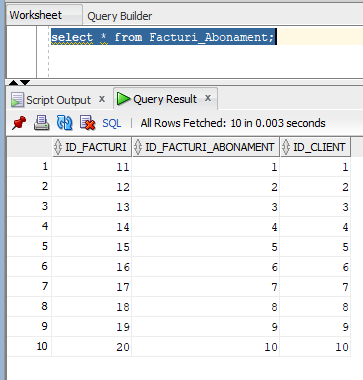
INSERT INTO Facturi\_Abonament (id\_facturi\_abonament, id\_client, id\_facturi) VALUES (7, 7, 17);

INSERT INTO Facturi\_Abonament (id\_facturi\_abonament, id\_client, id\_facturi) VALUES (8, 8, 18);

INSERT INTO Facturi\_Abonament (id\_facturi\_abonament, id\_client, id\_facturi) VALUES (9, 9, 19);

INSERT INTO Facturi\_Abonament (id\_facturi\_abonament, id\_client, id\_facturi) VALUES (10, 10, 20);





FACTURI\_PRODUSE:

* id\_facturi\_produse (NUMBER, PK): identificatorul unic al facturii pentru produse
* id\_achizitie (NUMBER, FK): cheia externă către tabela Achizitie

INSERT INTO Facturi\_Produse (id\_facturi\_produse, id\_facturi, id\_achizitie) VALUES (1, 21, 1);

INSERT INTO Facturi\_Produse (id\_facturi\_produse, id\_facturi, id\_achizitie) VALUES (2, 22, 2);

INSERT INTO Facturi\_Produse (id\_facturi\_produse, id\_facturi, id\_achizitie) VALUES (3, 23, 3);

INSERT INTO Facturi\_Produse (id\_facturi\_produse, id\_facturi, id\_achizitie) VALUES (4, 24, 4);

INSERT INTO Facturi\_Produse (id\_facturi\_produse, id\_facturi, id\_achizitie) VALUES (5, 25, 5);

INSERT INTO Facturi\_Produse (id\_facturi\_produse, id\_facturi, id\_achizitie) VALUES (6, 26, 6);

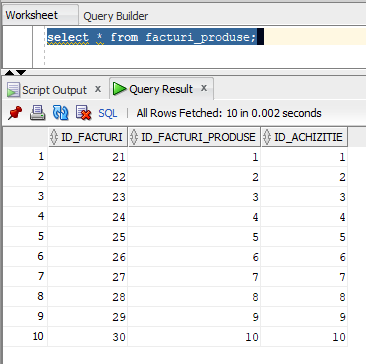
INSERT INTO Facturi\_Produse (id\_facturi\_produse, id\_facturi, id\_achizitie) VALUES (7, 27, 7);

INSERT INTO Facturi\_Produse (id\_facturi\_produse, id\_facturi, id\_achizitie) VALUES (8, 28, 8);

INSERT INTO Facturi\_Produse (id\_facturi\_produse, id\_facturi, id\_achizitie) VALUES (9, 29, 9);

INSERT INTO Facturi\_Produse (id\_facturi\_produse, id\_facturi, id\_achizitie) VALUES (10, 30, 10);





*6. 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.*

*Creați un subprogram stocat independent care primește de la tastatura id-ul unei săli, și afișează printr-o colecție toate aparatele pe care le tine. Tratați cazul în care id-ul sălii este invalid, sau aceasta nu are aparate.*

Prima data mi-am definit un tip de date (ca obiect), apoi un tip de date tabel, care este de tipul obiectului.

Am făcut acest lucru pentru ca să-i dau funcției parametrul de return. Iar pentru ca voi lucra cu obiecte, trebuie sa apelez constructor!!

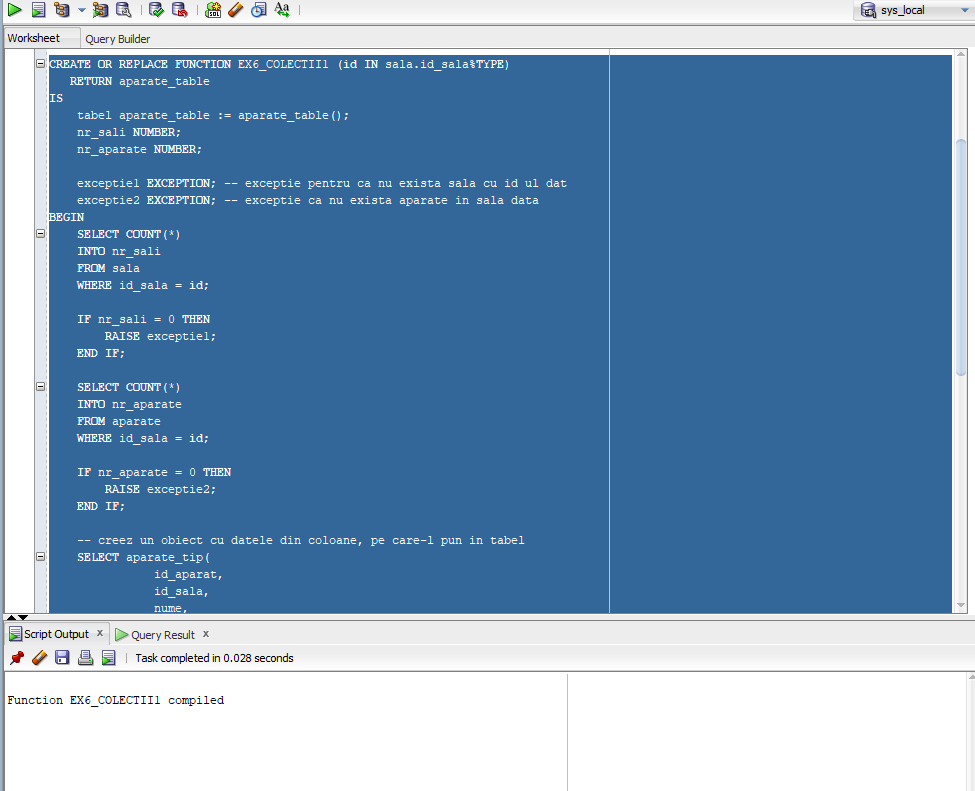
| CREATE OR REPLACE TYPE aparate\_tip AS OBJECT (  id\_aparat NUMBER,  id\_sala NUMBER,  nume VARCHAR2(100),  masa\_maxima NUMBER,  disponibilitate VARCHAR2(5),  grupa\_musculara\_lucrata VARCHAR2(255)  );  CREATE OR REPLACE TYPE aparate\_table AS TABLE OF aparate\_tip; |
| --- |

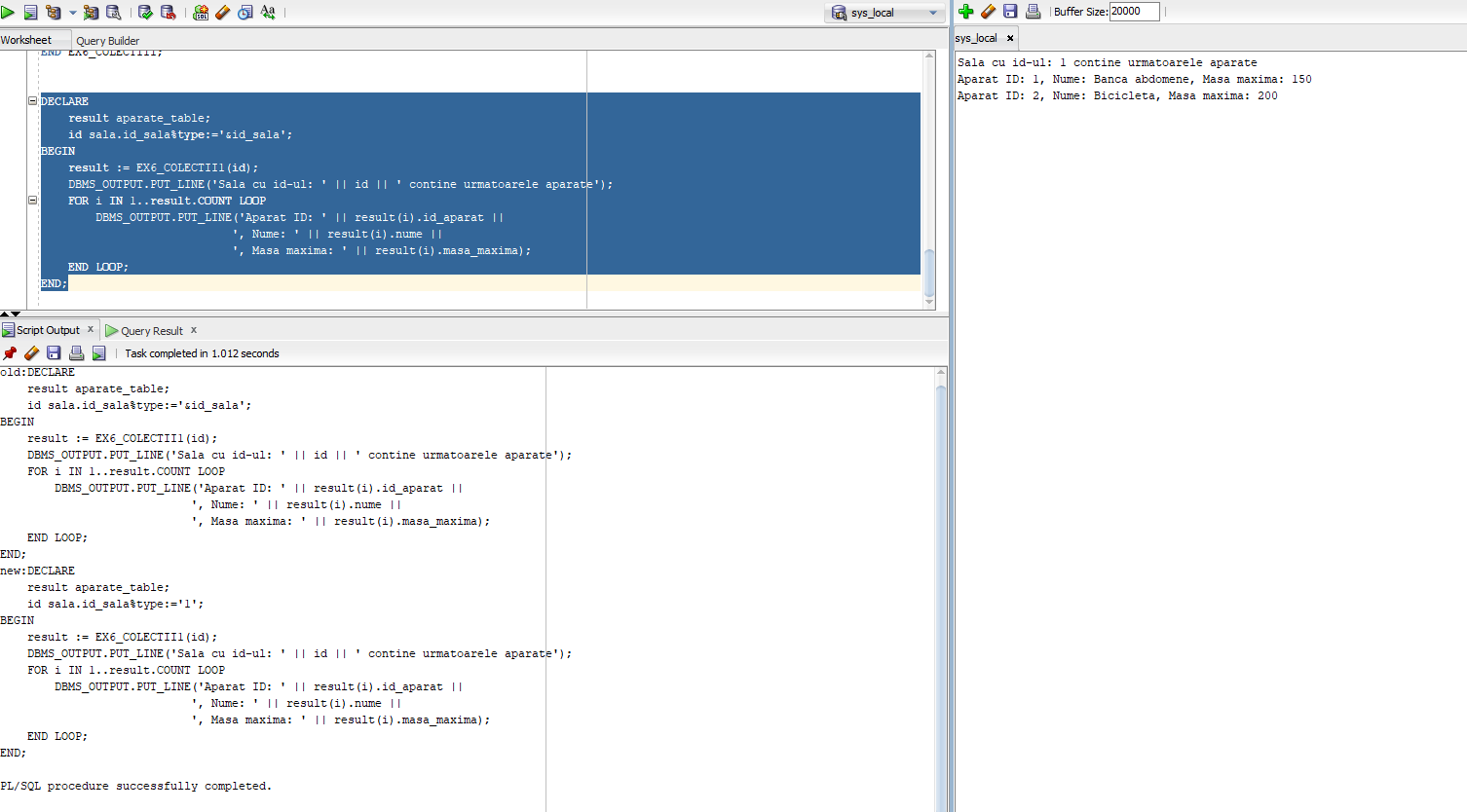
Tabel imbricat:

| CREATE OR REPLACE FUNCTION EX6\_COLECTII1 (id IN sala.id\_sala%TYPE)  RETURN aparate\_table  IS  tabel aparate\_table := aparate\_table();  nr\_sali NUMBER;  nr\_aparate NUMBER;  exceptie1 EXCEPTION; -- exceptie pentru ca nu exista sala cu id ul dat  exceptie2 EXCEPTION; -- exceptie ca nu exista aparate in sala data  BEGIN  SELECT COUNT(\*)  INTO nr\_sali  FROM sala  WHERE id\_sala = id;  IF nr\_sali = 0 THEN  RAISE exceptie1;  END IF;  SELECT COUNT(\*)  INTO nr\_aparate  FROM aparate  WHERE id\_sala = id;  IF nr\_aparate = 0 THEN  RAISE exceptie2;  END IF;    -- creez un obiect cu datele din coloane, pe care-l pun in tabel  SELECT aparate\_tip(  id\_aparat,  id\_sala,  nume,  masa\_maxima,  disponibilitate,  grupa\_musculara\_lucrata  )  BULK COLLECT INTO tabel  FROM aparate  WHERE id\_sala = id;  RETURN tabel;  EXCEPTION  WHEN exceptie1 THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista sala cu ID-ul: ' || id);  WHEN exceptie2 THEN  DBMS\_OUTPUT.PUT\_LINE('Sala cu ID-ul: ' || id || ' nu are aparate.');  END EX6\_COLECTII1; |
| --- |

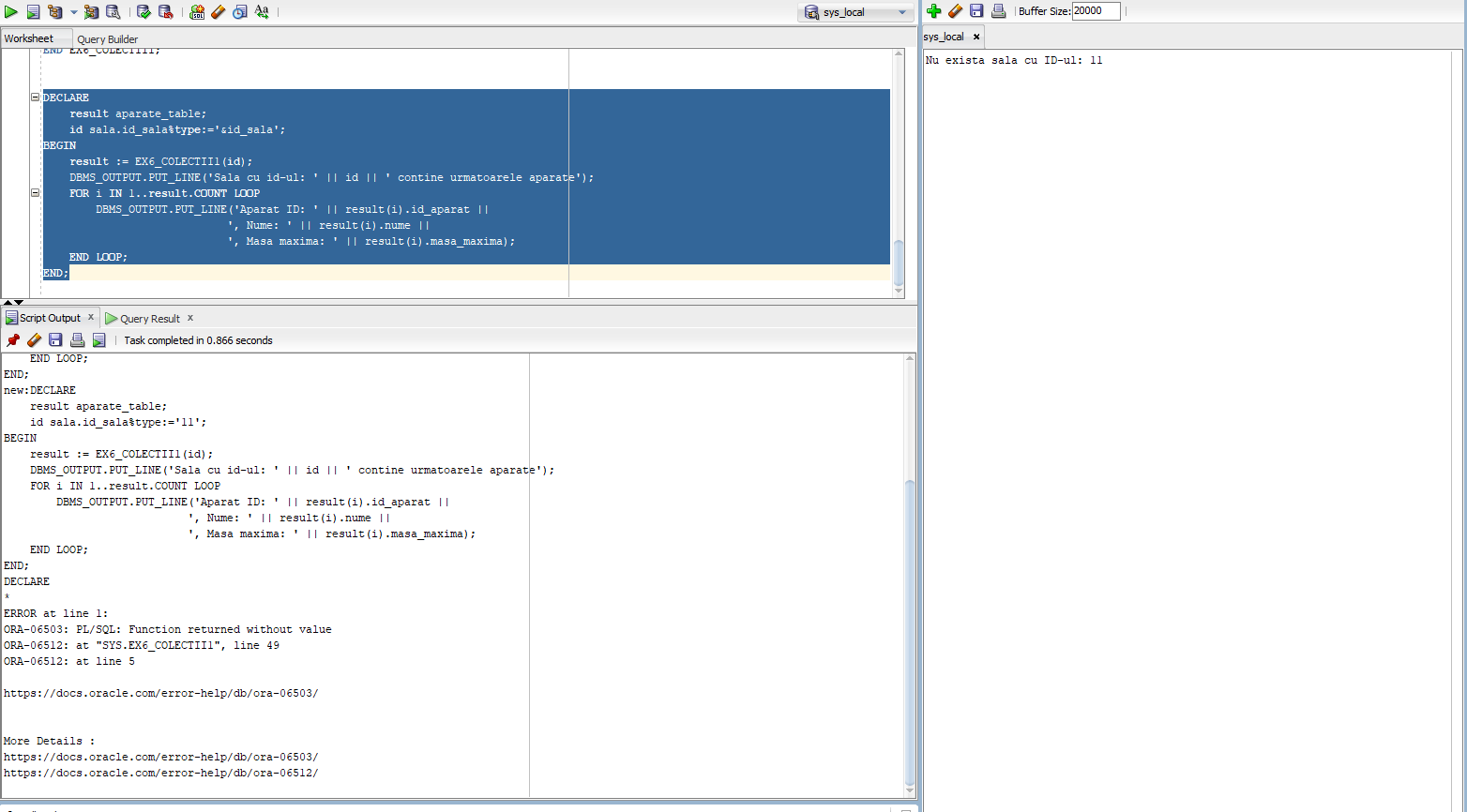
Blocul principal din care apelez functia.

| DECLARE  result aparate\_table;  id sala.id\_sala%type:='&id\_sala';  BEGIN  result := EX6\_COLECTII1(id);  DBMS\_OUTPUT.PUT\_LINE('Sala cu id-ul: ' || id || ' contine urmatoarele aparate');  FOR i IN 1..result.COUNT LOOP  DBMS\_OUTPUT.PUT\_LINE('Aparat ID: ' || result(i).id\_aparat ||  ', Nume: ' || result(i).nume ||  ', Masa maxima: ' || result(i).masa\_maxima);  END LOOP;  END; |
| --- |

Poza cu codul compilat:  
  
Apel pentru id sala: 1 (Un id valid care are aparate in ea:)



Acum pentru id 11, care nu este un id valid in baza de date pentru sala:

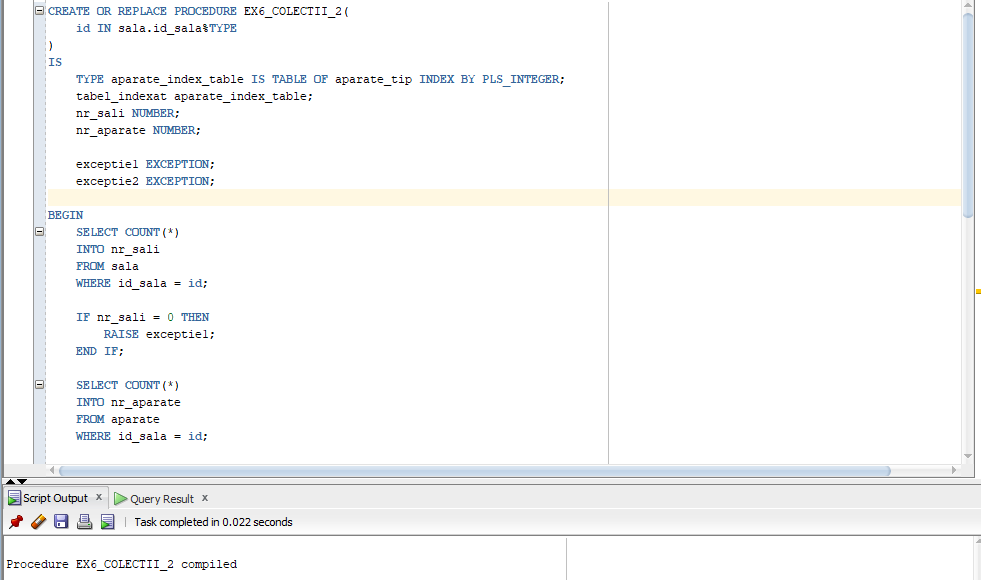


Iar acum pentru id 9, un id valid de sala, dar care nu are aparate in ea:



Tabel indexat:

| CREATE OR REPLACE PROCEDURE EX6\_COLECTII\_2(  id IN sala.id\_sala%TYPE  )  IS  TYPE aparate\_index\_table IS TABLE OF aparate\_tip INDEX BY PLS\_INTEGER;  tabel\_indexat aparate\_index\_table;  nr\_sali NUMBER;  nr\_aparate NUMBER;    exceptie1 EXCEPTION;  exceptie2 EXCEPTION;    BEGIN  SELECT COUNT(\*)  INTO nr\_sali  FROM sala  WHERE id\_sala = id;    IF nr\_sali = 0 THEN  RAISE exceptie1;  END IF;    SELECT COUNT(\*)  INTO nr\_aparate  FROM aparate  WHERE id\_sala = id;    IF nr\_aparate = 0 THEN  RAISE exceptie2;  END IF;    DECLARE  idx PLS\_INTEGER := 1;  BEGIN  FOR r IN (  SELECT id\_aparat, id\_sala, nume, masa\_maxima, disponibilitate, grupa\_musculara\_lucrata  FROM aparate  WHERE id\_sala = id  ) LOOP  tabel\_indexat(idx) := aparate\_tip(r.id\_aparat, r.id\_sala, r.nume, r.masa\_maxima, r.disponibilitate, r.grupa\_musculara\_lucrata);  idx := idx + 1;  END LOOP;  END;    DBMS\_OUTPUT.PUT\_LINE('Aparate in tabelul indexat:');  FOR i IN 1..tabel\_indexat.COUNT LOOP  DBMS\_OUTPUT.PUT\_LINE('Aparat ID: ' || tabel\_indexat(i).id\_aparat ||  ', Nume: ' || tabel\_indexat(i).nume ||  ', Masa maxima: ' || tabel\_indexat(i).masa\_maxima);  END LOOP;    EXCEPTION  WHEN exceptie1 THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista sala cu ID-ul: ' || id);  WHEN exceptie2 THEN  DBMS\_OUTPUT.PUT\_LINE('Sala cu ID-ul: ' || id || ' nu are aparate.');  END EX6\_COLECTII\_2; |
| --- |

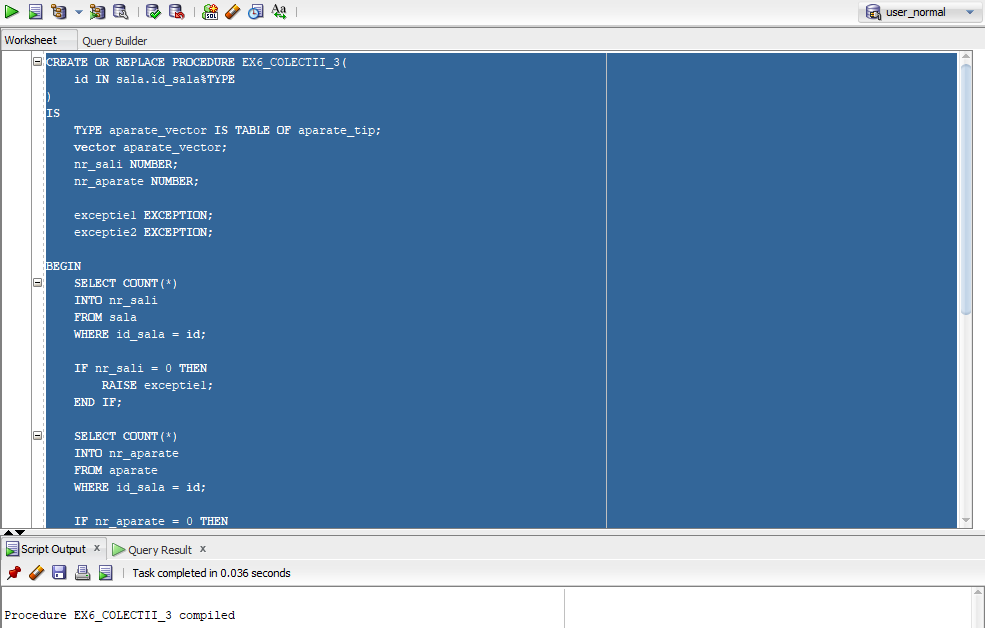


Blocul principal din care apelez functia:

| DECLARE  id sala.id\_sala%type:='&id\_sala';  BEGIN  EX6\_COLECTII\_2(id);  END; |
| --- |

Vector:

| CREATE OR REPLACE PROCEDURE EX6\_COLECTII\_3(  id IN sala.id\_sala%TYPE  )  IS  TYPE aparate\_vector IS TABLE OF aparate\_tip;  vector aparate\_vector;  nr\_sali NUMBER;  nr\_aparate NUMBER;    exceptie1 EXCEPTION;  exceptie2 EXCEPTION;    BEGIN  SELECT COUNT(\*)  INTO nr\_sali  FROM sala  WHERE id\_sala = id;    IF nr\_sali = 0 THEN  RAISE exceptie1;  END IF;    SELECT COUNT(\*)  INTO nr\_aparate  FROM aparate  WHERE id\_sala = id;    IF nr\_aparate = 0 THEN  RAISE exceptie2;  END IF;    SELECT aparate\_tip(id\_aparat, id\_sala, nume, masa\_maxima, disponibilitate, grupa\_musculara\_lucrata)  BULK COLLECT INTO vector  FROM aparate  WHERE id\_sala = id;    DBMS\_OUTPUT.PUT\_LINE('Aparate în vector:');  FOR i IN 1..vector.COUNT LOOP  DBMS\_OUTPUT.PUT\_LINE('Aparat ID: ' || vector(i).id\_aparat ||  ', Nume: ' || vector(i).nume ||  ', Masă maximă: ' || vector(i).masa\_maxima);  END LOOP;    EXCEPTION  WHEN exceptie1 THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista sala cu ID-ul: ' || id);  WHEN exceptie2 THEN  DBMS\_OUTPUT.PUT\_LINE('Sala cu ID-ul: ' || id || ' nu are aparate.');  END EX6\_COLECTII\_3; |
| --- |



Blocul din care este apelată procedura:

| DECLARE  id sala.id\_sala%type:='&id\_sala';  BEGIN  EX6\_COLECTII\_3(id);  END; |
| --- |

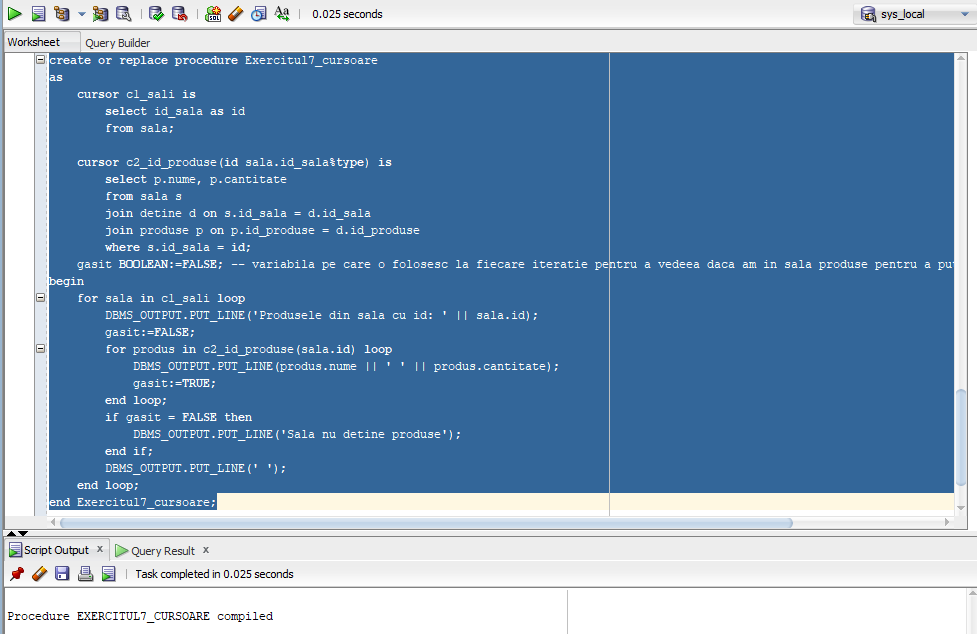
*7.Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor. Apelați subprogramul.*

Definiți un subprogram care utilizează 2 cursoare, unul care returnează id-ul fiecărei săli, iar cel de-al doilea, un cursor parametrizat, care primește ca parametru id-ul unei săli și returnează produsele din aceasta sala. În cazul în care sala nu deține niciun produs, afișați un mesaj specific.

Ciclu cursor:

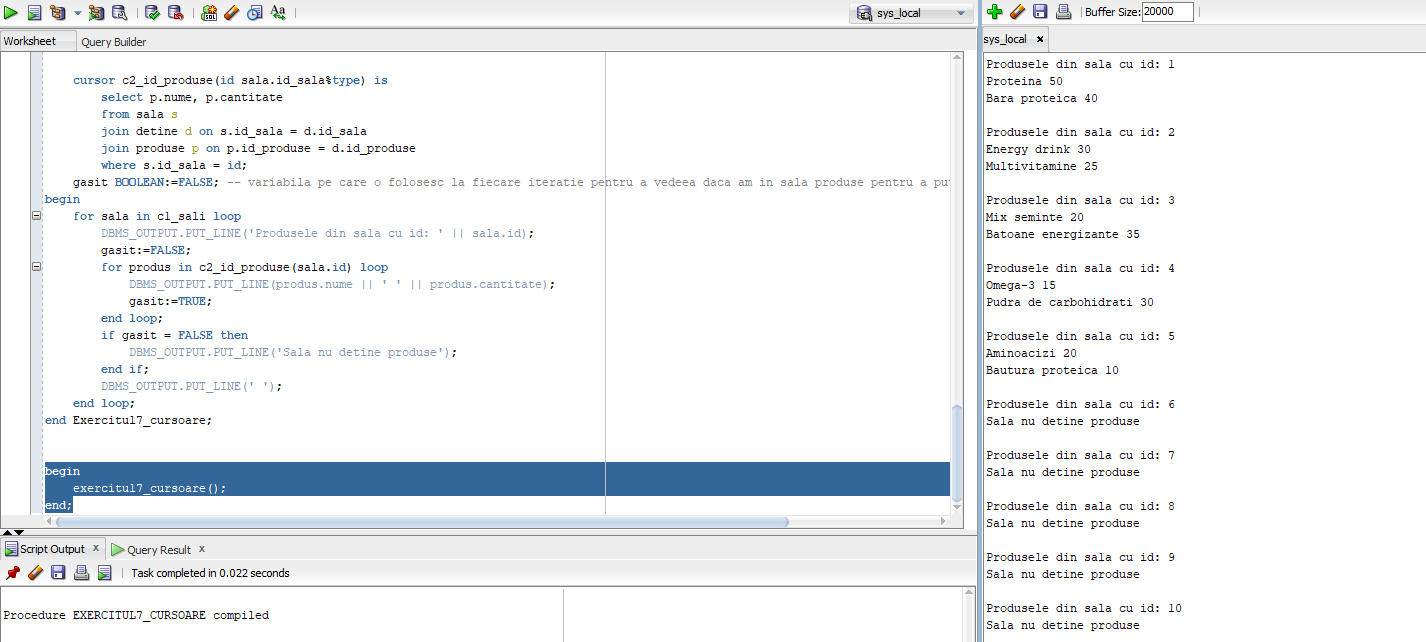
Procedura:

| create or replace procedure Exercitul7\_cursoare  as  cursor c1\_sali is  select id\_sala as id  from sala;  cursor c2\_id\_produse(id sala.id\_sala%type) is  select p.nume, p.cantitate  from sala s  join detine d on s.id\_sala = d.id\_sala  join produse p on p.id\_produse = d.id\_produse  where s.id\_sala = id;  gasit BOOLEAN:=FALSE; -- variabila pe care o folosesc la fiecare iteratie pentru a vedea daca am in sala produse pentru a putea da un mesaj specific.  begin  for sala in c1\_sali loop  DBMS\_OUTPUT.PUT\_LINE('Produsele din sala cu id: ' || sala.id);  gasit:=FALSE;  for produs in c2\_id\_produse(sala.id) loop  DBMS\_OUTPUT.PUT\_LINE(produs.nume || ' ' || produs.cantitate);  gasit:=TRUE;  end loop;  if gasit = FALSE then  DBMS\_OUTPUT.PUT\_LINE('Sala nu detine produse');  end if;  DBMS\_OUTPUT.PUT\_LINE(' ');  end loop;  end Exercitul7\_cursoare; |
| --- |



Blocul principal din care apelez procedura:

| begin  exercitul7\_cursoare();  end; |
| --- |

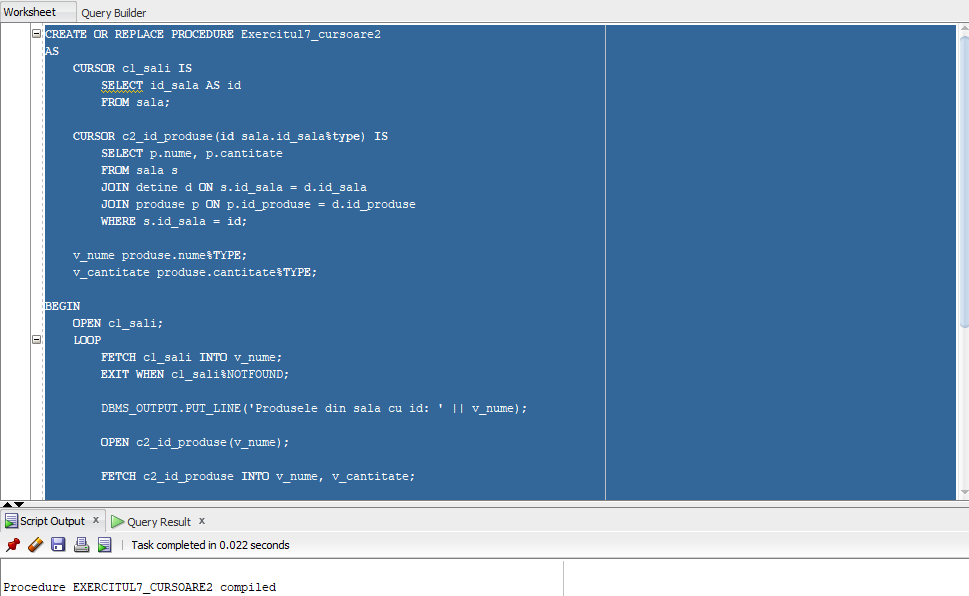


O alta varianta a acestei proceduri este prin definirea cursorilor cu open + fetch:

Varianta 2:

OPEN + Fetch:

| CREATE OR REPLACE PROCEDURE Exercitul7\_cursoare2  AS  CURSOR c1\_sali IS  SELECT id\_sala AS id  FROM sala;  CURSOR c2\_id\_produse(id sala.id\_sala%type) IS  SELECT p.nume, p.cantitate  FROM sala s  JOIN detine d ON s.id\_sala = d.id\_sala  JOIN produse p ON p.id\_produse = d.id\_produse  WHERE s.id\_sala = id;  v\_nume produse.nume%TYPE;  v\_cantitate produse.cantitate%TYPE;  BEGIN  OPEN c1\_sali;  LOOP  FETCH c1\_sali INTO v\_nume;  EXIT WHEN c1\_sali%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE('Produsele din sala cu id: ' || v\_nume);  OPEN c2\_id\_produse(v\_nume);    FETCH c2\_id\_produse INTO v\_nume, v\_cantitate;  IF c2\_id\_produse%NOTFOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Sala nu detine produse');  ELSE  LOOP  DBMS\_OUTPUT.PUT\_LINE(v\_nume || ' ' || v\_cantitate);  FETCH c2\_id\_produse INTO v\_nume, v\_cantitate;  EXIT WHEN c2\_id\_produse%NOTFOUND;  END LOOP;  END IF;  CLOSE c2\_id\_produse;  DBMS\_OUTPUT.PUT\_LINE(' ');  END LOOP;  CLOSE c1\_sali;  END Exercitul7\_cursoare2; |
| --- |



Bloc de apel:

| begin  Exercitul7\_cursoare2();  end; |
| --- |

*8.Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele create. Tratați toate excepțiile care pot apărea, incluzând excepțiile predefinite NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.*

Creați un subprogram stocat independent de tip funcție care să obțină pentru un id de sala, dat ca parametru, informații despre clienți, respectiv despre abonamentele pe care aceștia le dețin, dacă au expirat sau nu.

Tratați cazurile în care:

* Id-ul sălii nu corespunde cu nicio sala existentă
* Sala in cauza nu are clienti (prin abonamente)
* În sală respectivă exista 2 clienți cu nume identic.

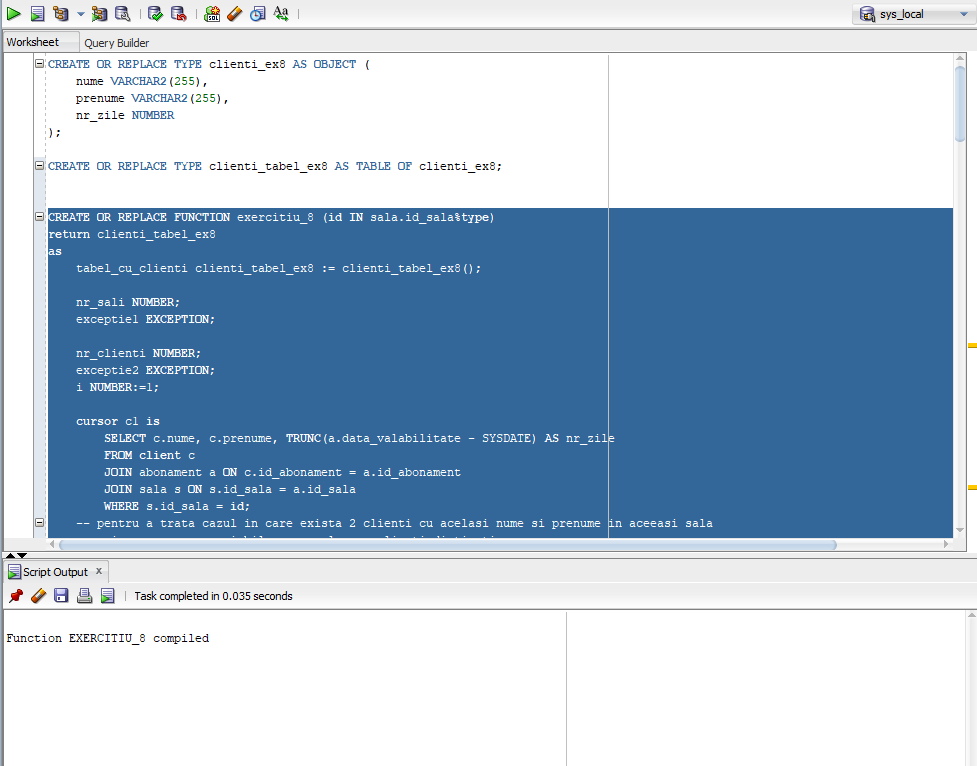
Pentru acest exercitiu, mi-am definit înainte tipurile de date pe care le voi folosii:

| CREATE OR REPLACE TYPE clienti\_ex8 AS OBJECT (  nume VARCHAR2(255),  prenume VARCHAR2(255),  nr\_zile NUMBER  );  CREATE OR REPLACE TYPE clienti\_tabel\_ex8 AS TABLE OF clienti\_ex8; |
| --- |

Îmi propun sa returnez un tabel cu clienții care aparțin de acea sală (printr-un anume abonament);

| CREATE OR REPLACE FUNCTION exercitiu\_8 (id IN sala.id\_sala%type)  return clienti\_tabel\_ex8  as  tabel\_cu\_clienti clienti\_tabel\_ex8 := clienti\_tabel\_ex8();  nr\_sali NUMBER;  exceptie1 EXCEPTION;    nr\_clienti NUMBER;  exceptie2 EXCEPTION;  i NUMBER:=1;    cursor c1 is  SELECT c.nume, c.prenume, TRUNC(a.data\_valabilitate - SYSDATE) AS nr\_zile  FROM client c  JOIN abonament a ON c.id\_abonament = a.id\_abonament  JOIN sala s ON s.id\_sala = a.id\_sala  WHERE s.id\_sala = id;  -- pentru a trata cazul in care exista 2 clienti cu acelasi nume si prenume in aceeasi sala  -- voi crea o noua variabila cu numele: nr\_clienti\_distincti  -- aceasta numara cu un count distinct, numarul de clienti unic, care au combinatia nume + prenume  -- daca sunt 2 cu acelasi nume, count distinct returneaza 1,  -- dar in tabel voi avea 2    -- deci, voi compara valoarea din nr\_clienti\_distincti cu nr de clineti din tabel, daca e egal nu am prob, daca nu, exceptie;    nr\_clienti\_distincti NUMBER;  exceptie3 EXCEPTION;  BEGIN  SELECT COUNT(id\_sala)  INTO nr\_sali  FROM sala  WHERE id\_sala = id;    IF nr\_sali = 0 THEN  RAISE exceptie1;  END IF;  SELECT COUNT(c.id\_client)  INTO nr\_clienti  FROM client c  JOIN abonament a ON c.id\_abonament = a.id\_abonament  JOIN sala s ON s.id\_sala = a.id\_sala  WHERE s.id\_sala = id;    select count(distinct(c.nume || c.prenume))  into nr\_clienti\_distincti  FROM client c  JOIN abonament a ON c.id\_abonament = a.id\_abonament  JOIN sala s ON s.id\_sala = a.id\_sala  WHERE s.id\_sala = id;    IF nr\_clienti = 0 THEN  RAISE exceptie2;  END IF;    for client in c1 loop  tabel\_cu\_clienti.EXTEND;  tabel\_cu\_clienti(i):= clienti\_ex8(client.nume, client.prenume, client.nr\_zile);  i:= i+1;  end loop;  if nr\_clienti\_distincti = tabel\_cu\_clienti.count() then  return tabel\_cu\_clienti;  else  raise exceptie3;  end if;  EXCEPTION  WHEN exceptie1 THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista sala data');  return clienti\_tabel\_ex8();  WHEN exceptie2 THEN  DBMS\_OUTPUT.PUT\_LINE('Sala data nu are clienti');  return clienti\_tabel\_ex8();  when exceptie3 then  DBMS\_OUTPUT.PUT\_LINE('In sala exista cel putin 2 clienti cu acelasi nume');  return clienti\_tabel\_ex8();  END exercitiu\_8; |
| --- |

Poza ca funcție a compilat:



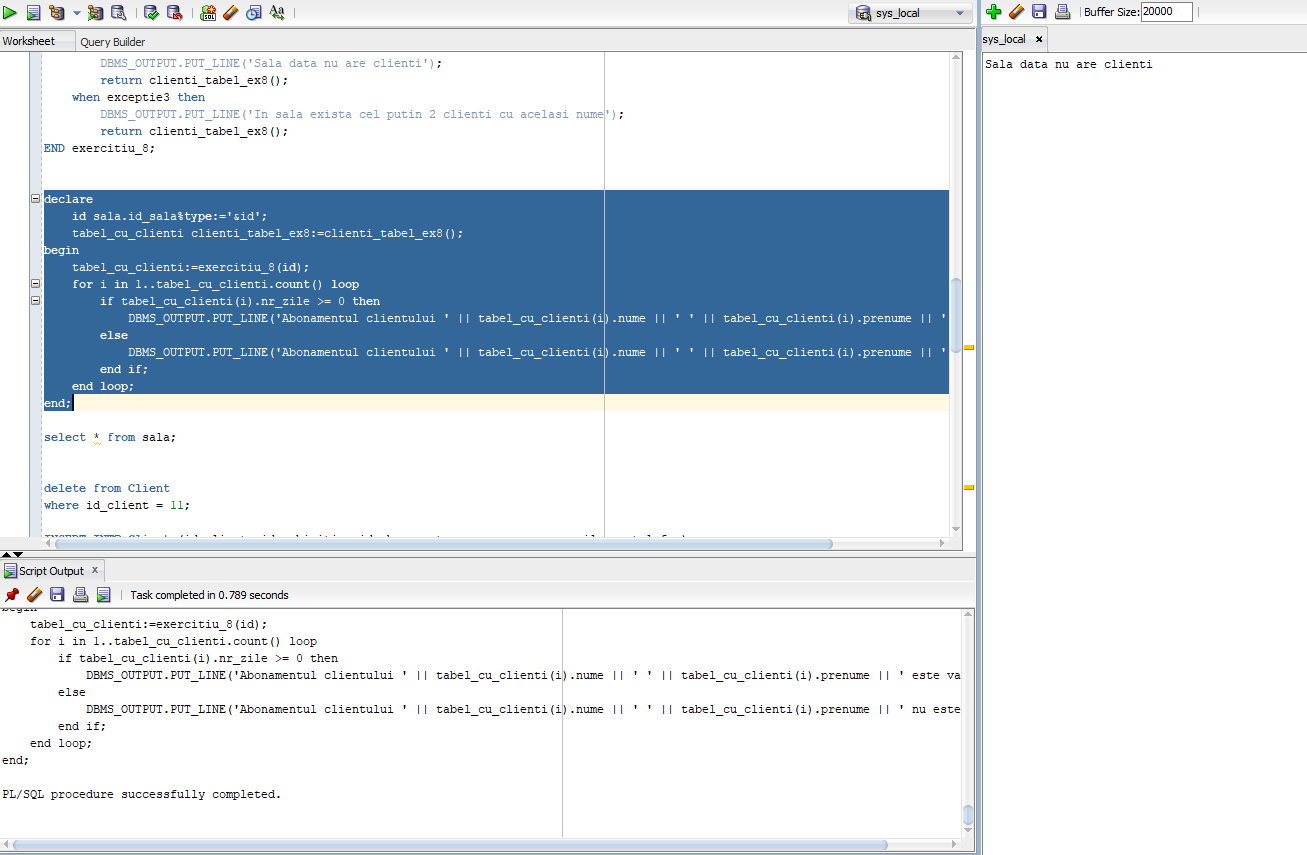
Blocul din care apelez funcția:

| declare  id sala.id\_sala%type:='&id';  tabel\_cu\_clienti clienti\_tabel\_ex8:=clienti\_tabel\_ex8();  begin  tabel\_cu\_clienti:=exercitiu\_8(id);  for i in 1..tabel\_cu\_clienti.count() loop  if tabel\_cu\_clienti(i).nr\_zile >= 0 then  DBMS\_OUTPUT.PUT\_LINE('Abonamentul clientului ' || tabel\_cu\_clienti(i).nume || ' ' || tabel\_cu\_clienti(i).prenume || ' este valid');  else  DBMS\_OUTPUT.PUT\_LINE('Abonamentul clientului ' || tabel\_cu\_clienti(i).nume || ' ' || tabel\_cu\_clienti(i).prenume || ' nu este valid');  end if;  end loop;  end; |
| --- |

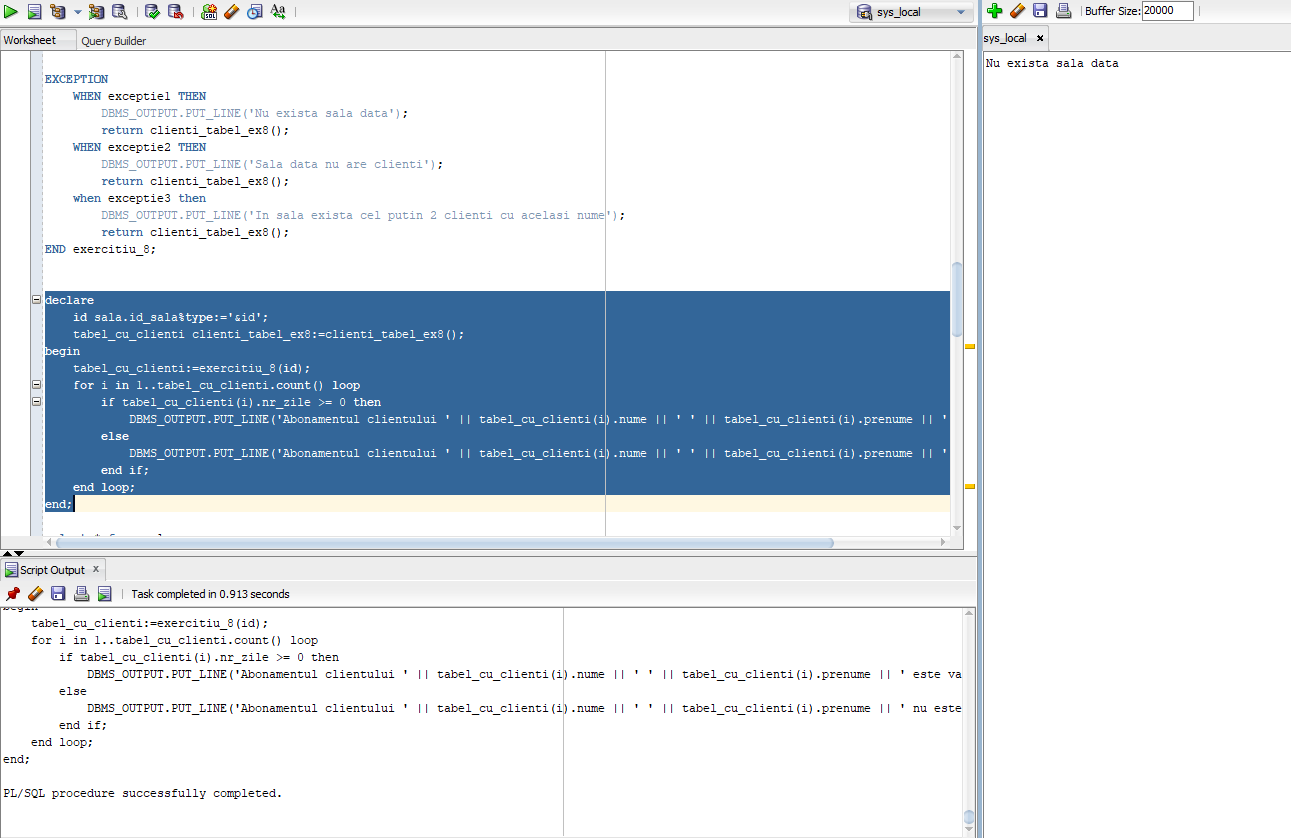
Rularea codului pentru id = 1:



Pentru id = 7 (sala care nu are clienti/abonamente):



Pentru id = 11: sala care nu exista



Și acum, pentru a exemplifica exceptia cu too\_many\_rows, voi insera un nou client (cu același nume și prenume ca unul dintr-o sală deja existentă);

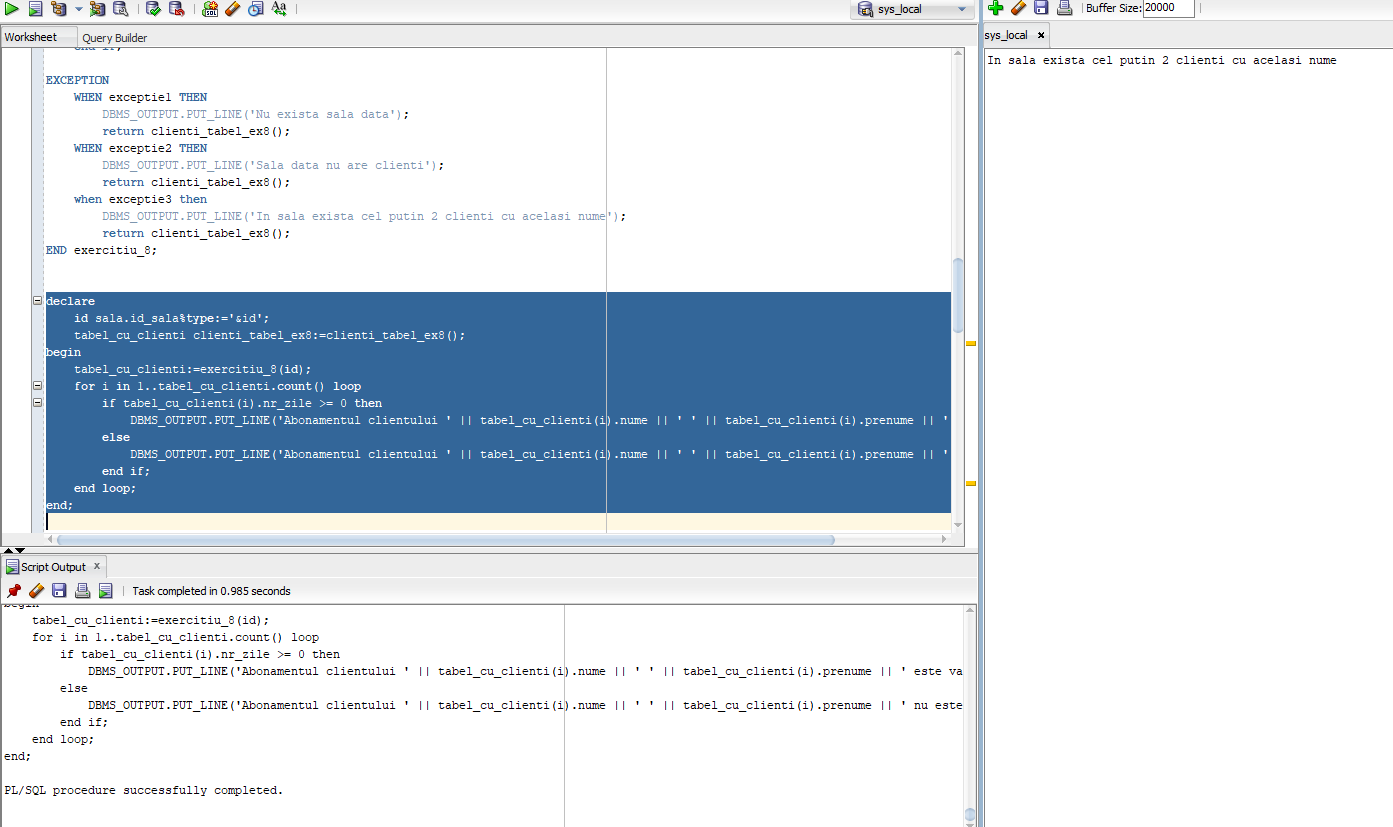
De exemplu, pentru sala cu id 2, avem următorii clienți:

Pop Ana și Georgescu Andrei

Astfel, voi mai insera inca o data, pe Pop Ana prin comanda:

| INSERT INTO Client (id\_client, id\_abonament, nume, prenume, email, nr\_telefon)  VALUES (11, 3, 'Pop', 'Ana', 'pop.ana@gmail.com', '0743123456'); |
| --- |

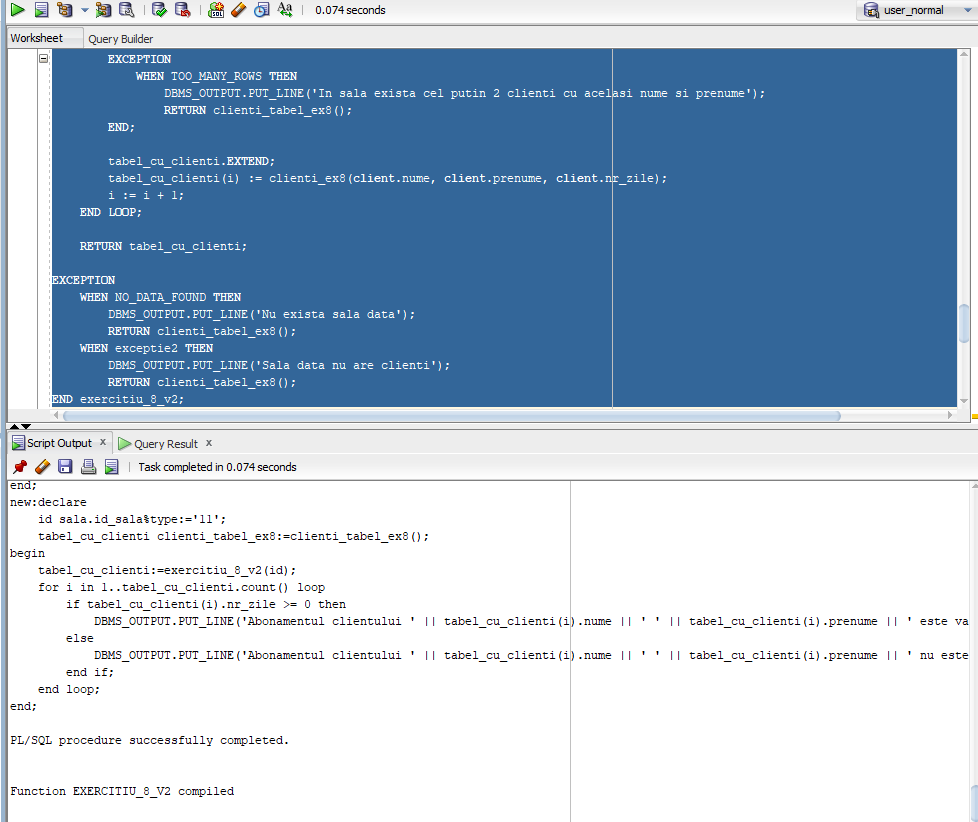
Iar acum cand apelez funcția cu id = 2, voi primi textul ca în sala am cel puțin 2 clienți cu același nume și prenume:



| (delete ul aferent pentru a restaura structura datelor: delete from Client  where id\_client = 11;  ) |
| --- |

Exemplific excepțiile 1 și 3 și prin cele predefinite din sistem (NO\_DATA\_FOUND și TOO\_MANY\_ROWS). Modul în care voi arata acest lucru este prin variabile de test, prin care voi face cate un select .. into variabila\_test. Mă voi folosii de v\_id\_sala pentru NO\_DATA\_FOUND și v\_nume pentru TOO\_MANY\_ROWS, în cazul în care exista 2 clienți cu același nume.

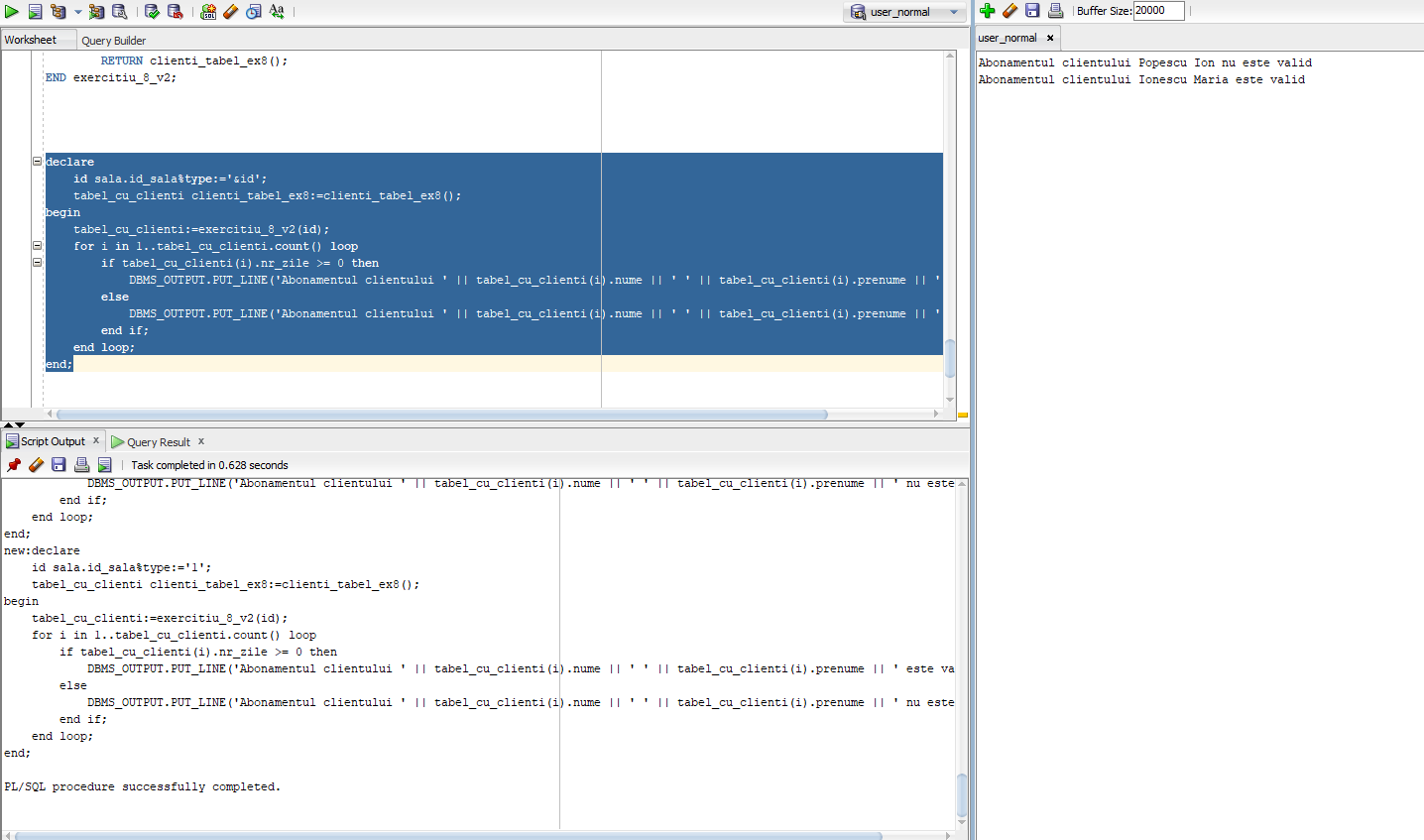
| CREATE OR REPLACE FUNCTION exercitiu\_8\_v2 (id IN sala.id\_sala%type)  RETURN clienti\_tabel\_ex8  AS  tabel\_cu\_clienti clienti\_tabel\_ex8 := clienti\_tabel\_ex8();  nr\_sali NUMBER;  nr\_clienti NUMBER;  exceptie2 EXCEPTION;  i NUMBER := 1;  cursor c1 is  SELECT c.nume, c.prenume, TRUNC(a.data\_valabilitate - SYSDATE) AS nr\_zile  FROM client c  JOIN abonament a ON c.id\_abonament = a.id\_abonament  JOIN sala s ON s.id\_sala = a.id\_sala  WHERE s.id\_sala = id;  v\_id\_sala NUMBER;  v\_nume client.nume%type;  BEGIN  -- verific daca exista sala  SELECT id\_sala  INTO v\_id\_sala  FROM sala  WHERE id\_sala = id;  -- calculez cu un count nr de clienti care apartin salii  SELECT COUNT(c.id\_client)  INTO nr\_clienti  FROM client c  JOIN abonament a ON c.id\_abonament = a.id\_abonament  JOIN sala s ON s.id\_sala = a.id\_sala  WHERE s.id\_sala = id;  IF nr\_clienti = 0 THEN  RAISE exceptie2;  END IF;  -- parcurg cursorul pentru a trata exceptia + a adauga clientii in tabel  FOR client IN c1 LOOP  -- verific daca exista clienti cu ac nume si prenume, ma folosesc de var v\_nume pentru a exemplifica TOO\_MANY\_ROWS  -- NO\_DATA\_FOUND nu e posibil in acest subloc deoarece deja a fost tratata mai sus.  BEGIN  SELECT c.nume  INTO v\_nume  FROM client c  WHERE c.nume = client.nume AND c.prenume = client.prenume;  EXCEPTION  WHEN TOO\_MANY\_ROWS THEN  DBMS\_OUTPUT.PUT\_LINE('In sala exista cel putin 2 clienti cu acelasi nume si prenume');  RETURN clienti\_tabel\_ex8();  END;  tabel\_cu\_clienti.EXTEND;  tabel\_cu\_clienti(i) := clienti\_ex8(client.nume, client.prenume, client.nr\_zile);  i := i + 1;  END LOOP;  RETURN tabel\_cu\_clienti;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Nu exista sala data');  RETURN clienti\_tabel\_ex8();  WHEN exceptie2 THEN  DBMS\_OUTPUT.PUT\_LINE('Sala data nu are clienti');  RETURN clienti\_tabel\_ex8();  END exercitiu\_8\_v2; |
| --- |



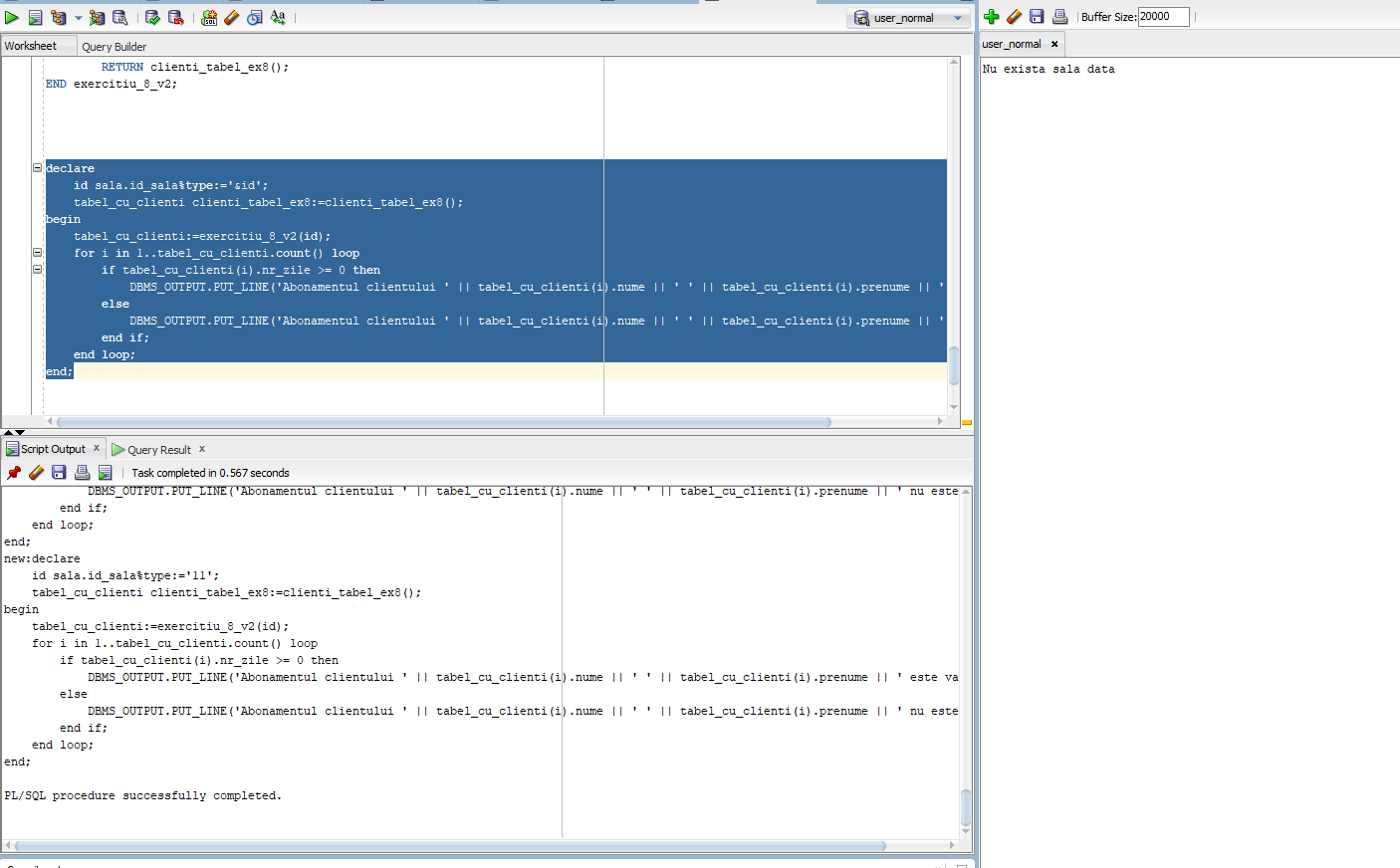
Blocul din care apelez funcția:

| declare  id sala.id\_sala%type:='&id';  tabel\_cu\_clienti clienti\_tabel\_ex8:=clienti\_tabel\_ex8();  begin  tabel\_cu\_clienti:=exercitiu\_8\_v2(id);  for i in 1..tabel\_cu\_clienti.count() loop  if tabel\_cu\_clienti(i).nr\_zile >= 0 then  DBMS\_OUTPUT.PUT\_LINE('Abonamentul clientului ' || tabel\_cu\_clienti(i).nume || ' ' || tabel\_cu\_clienti(i).prenume || ' este valid');  else  DBMS\_OUTPUT.PUT\_LINE('Abonamentul clientului ' || tabel\_cu\_clienti(i).nume || ' ' || tabel\_cu\_clienti(i).prenume || ' nu este valid');  end if;  end loop;  end; |
| --- |

Screenshot pentru id = 1: (Varianta 1 și 2 au fost făcute la date diferite, deci exista posibilitatea ca abonamentul unor clienți să fie expirat)

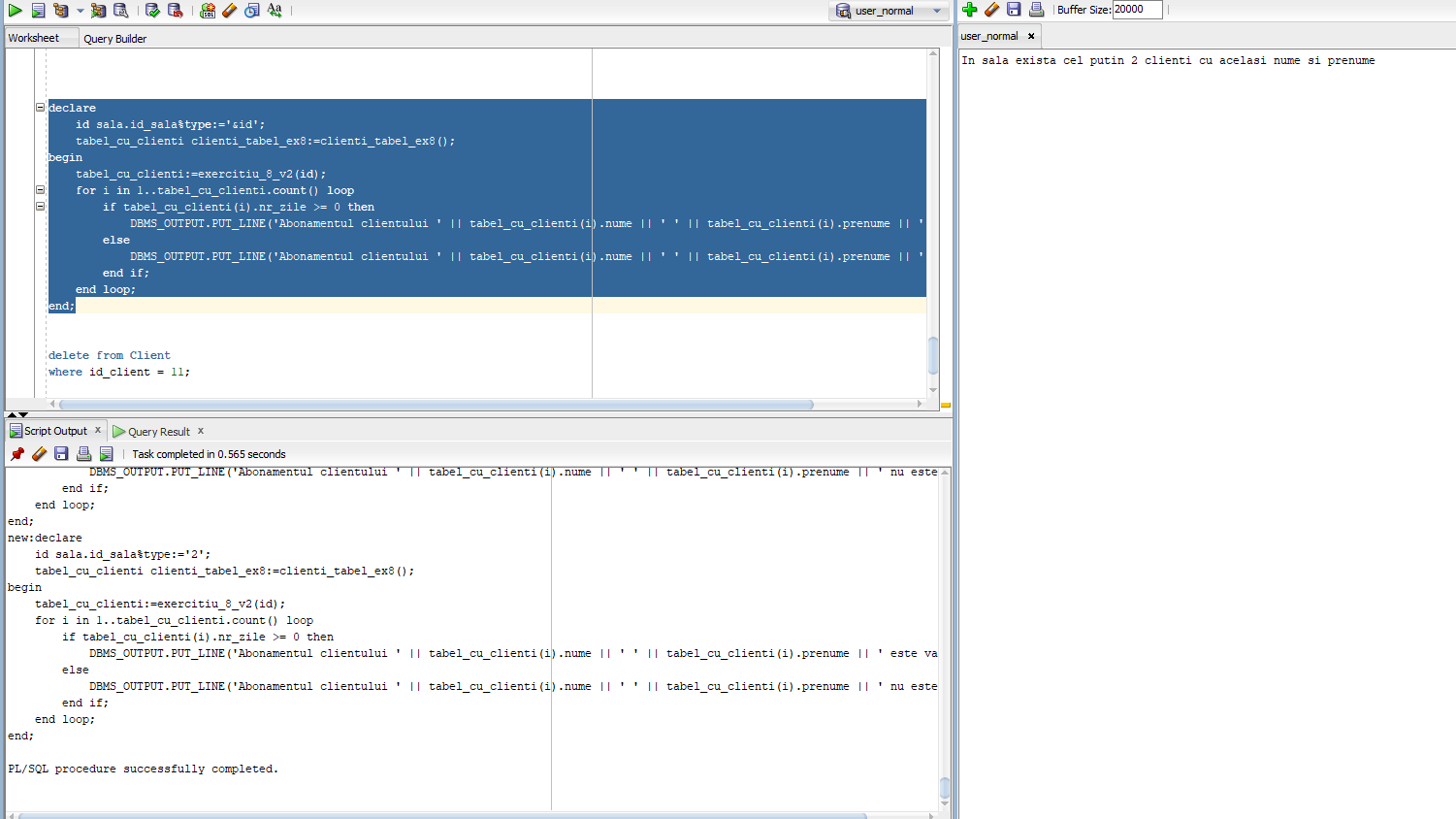


Screenshot pentru id = 11: (NO\_DATA\_FOUND):



Screenshot pentru id = 2 (Dar prima data fac insert ul sa am in sala cu id 2, doua persoane cu același nume și prenume).

| INSERT INTO Client (id\_client, id\_abonament, nume, prenume, email, nr\_telefon)  VALUES (11, 3, 'Pop', 'Ana', 'pop.ana@gmail.com', '0743123456'); |
| --- |



Și delete ul pentru a restaura structura tabelului Sala:

| delete from Client  where id\_client = 11; |
| --- |

*9.Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să aibă minim 2 parametri și să utilizeze într-o singură comandă SQL 5 dintre tabelele create. Definiți minim 2 excepții proprii, altele decât cele predefinite la nivel de sistem. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate.*

Pentru un ID de sala dat la tastatură, afișați numele clienților care au cumparat produse cu un preț mai mare decat un preț dat de la tastatură.

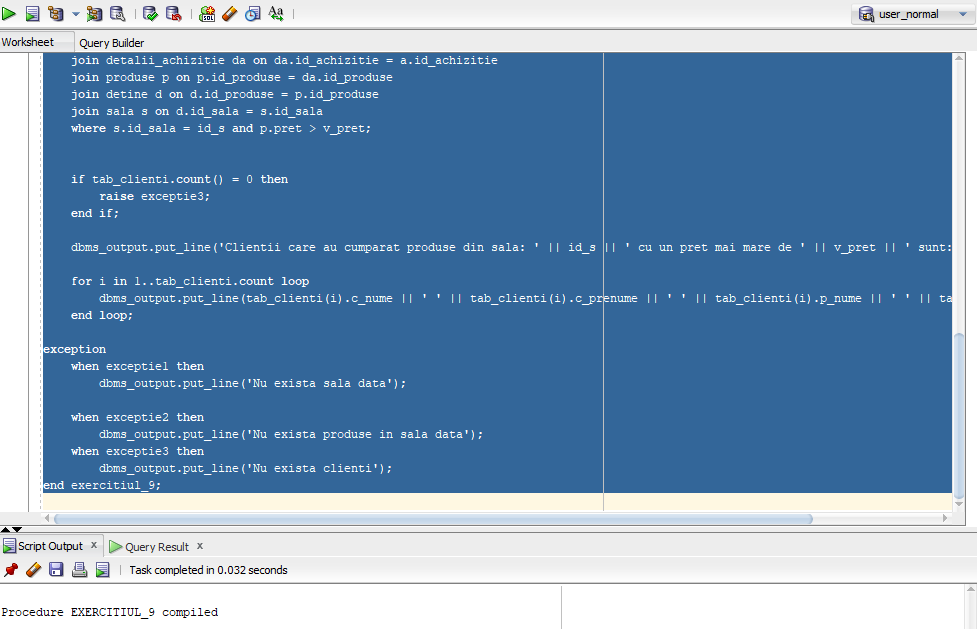
Sa se trateze cazurile în care:

* Sala data nu exista
* Sala data nu are produse
* Produsele existente nu au fost cumparate de niciun client.

Nu este cazul și pentru too\_many\_rows în cazul în care exista mai mulți clienți cu ac nume și prenume ca la exercițiul trecut, deoarece un client poate cumpara mai multe produse de mai multe ori.

Procedura:

| create or replace procedure exercitiul\_9(id\_s sala.id\_sala%type, v\_pret NUMBER)  AS  exceptie1 EXCEPTION; -- verificam daca exista sau nu sala data  nrSali NUMBER;    exceptie2 EXCEPTION; -- verificam daca exista sau nu produse in sala data  nrProduse NUMBER;    type t\_client is record(  c\_nume client.nume%type,  c\_prenume client.prenume%type,  p\_nume produse.nume%type,  p\_pret produse.pret%type  );    type tabel\_clienti is table of t\_client;  tab\_clienti tabel\_clienti;  exceptie3 EXCEPTION;  begin  select count(id\_sala)  into nrSali  from sala  where id\_sala = id\_s;    if nrSali = 0 then  raise exceptie1;  end if;    select count(p.id\_produse)  into nrProduse  from produse p  join detine d on d.id\_produse = p.id\_produse  join sala s on s.id\_sala = d.id\_sala  where s.id\_sala = id\_s;    if nrProduse = 0 then  raise exceptie2;  end if;    select c.nume, c.prenume, p.nume, p.pret  bulk collect into tab\_clienti  from client c  join achizitie a on c.id\_client = a.id\_client  join detalii\_achizitie da on da.id\_achizitie = a.id\_achizitie  join produse p on p.id\_produse = da.id\_produse  join detine d on d.id\_produse = p.id\_produse  join sala s on d.id\_sala = s.id\_sala  where s.id\_sala = id\_s and p.pret > v\_pret;      if tab\_clienti.count() = 0 then  raise exceptie3;  end if;    dbms\_output.put\_line('Clientii care au cumparat produse din sala: ' || id\_s || ' cu un pret mai mare de ' || v\_pret || ' sunt:');    for i in 1..tab\_clienti.count loop  dbms\_output.put\_line(tab\_clienti(i).c\_nume || ' ' || tab\_clienti(i).c\_prenume || ' ' || tab\_clienti(i).p\_nume || ' ' || tab\_clienti(i).p\_pret);  end loop;    exception  when exceptie1 then  dbms\_output.put\_line('Nu exista sala data');    when exceptie2 then  dbms\_output.put\_line('Nu exista produse in sala data');  when exceptie3 then  dbms\_output.put\_line('Nu exista clienti');  end exercitiul\_9; |
| --- |

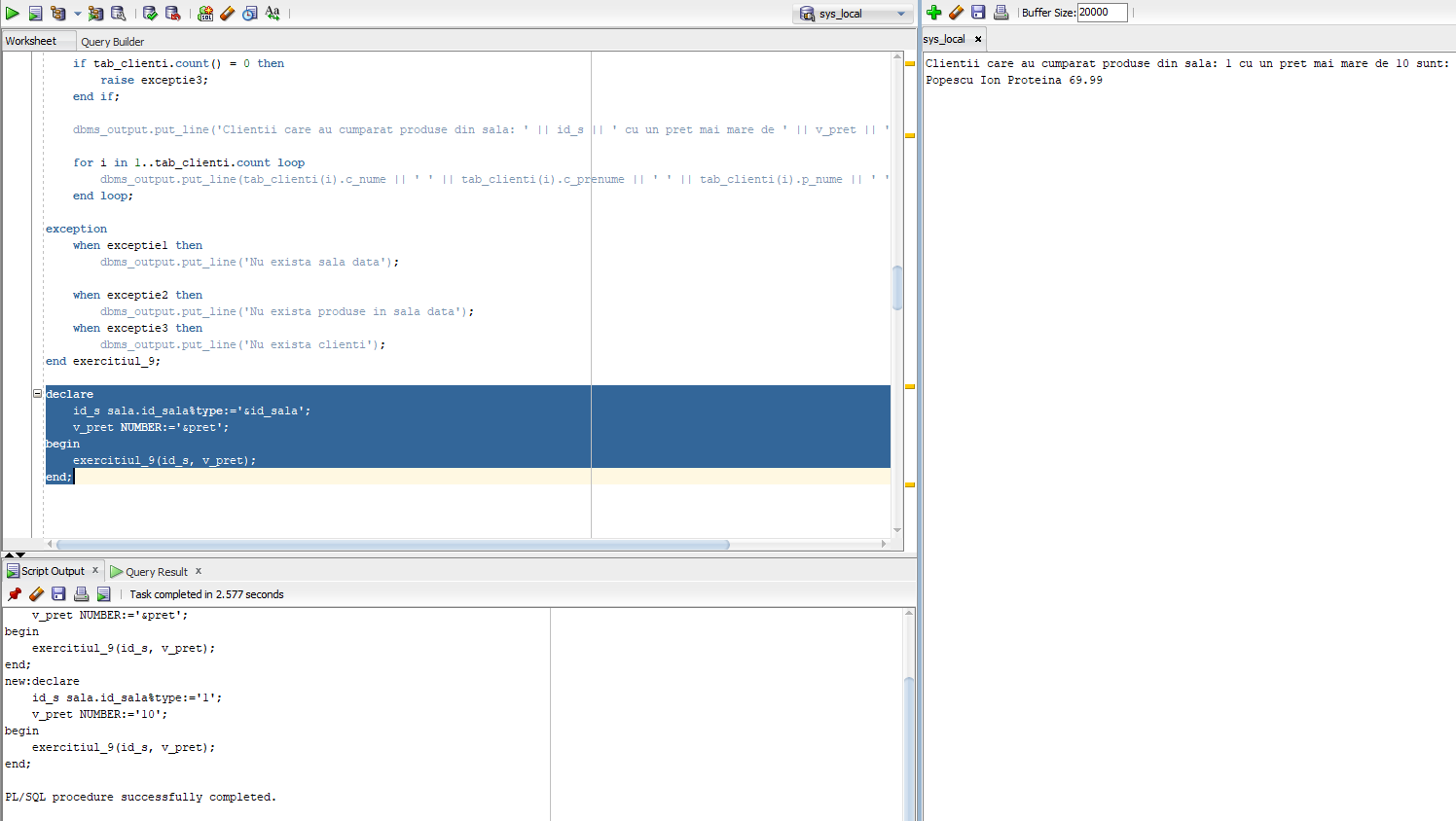


Blocul din care este apelată:

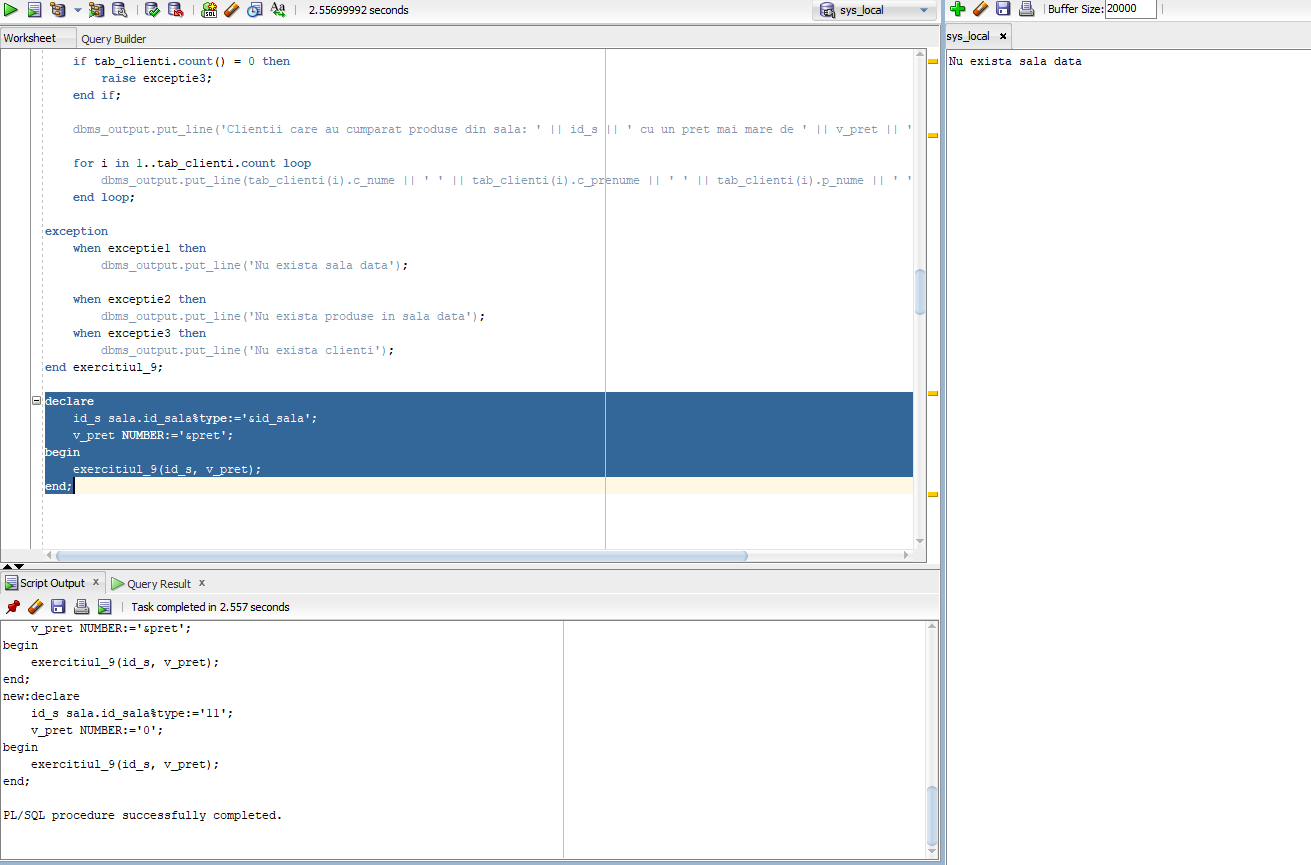
| declare  id\_s sala.id\_sala%type:='&id\_sala';  v\_pret NUMBER:='&pret';  begin  exercitiul\_9(id\_s, v\_pret);  end; |
| --- |

Apeluri pentru:

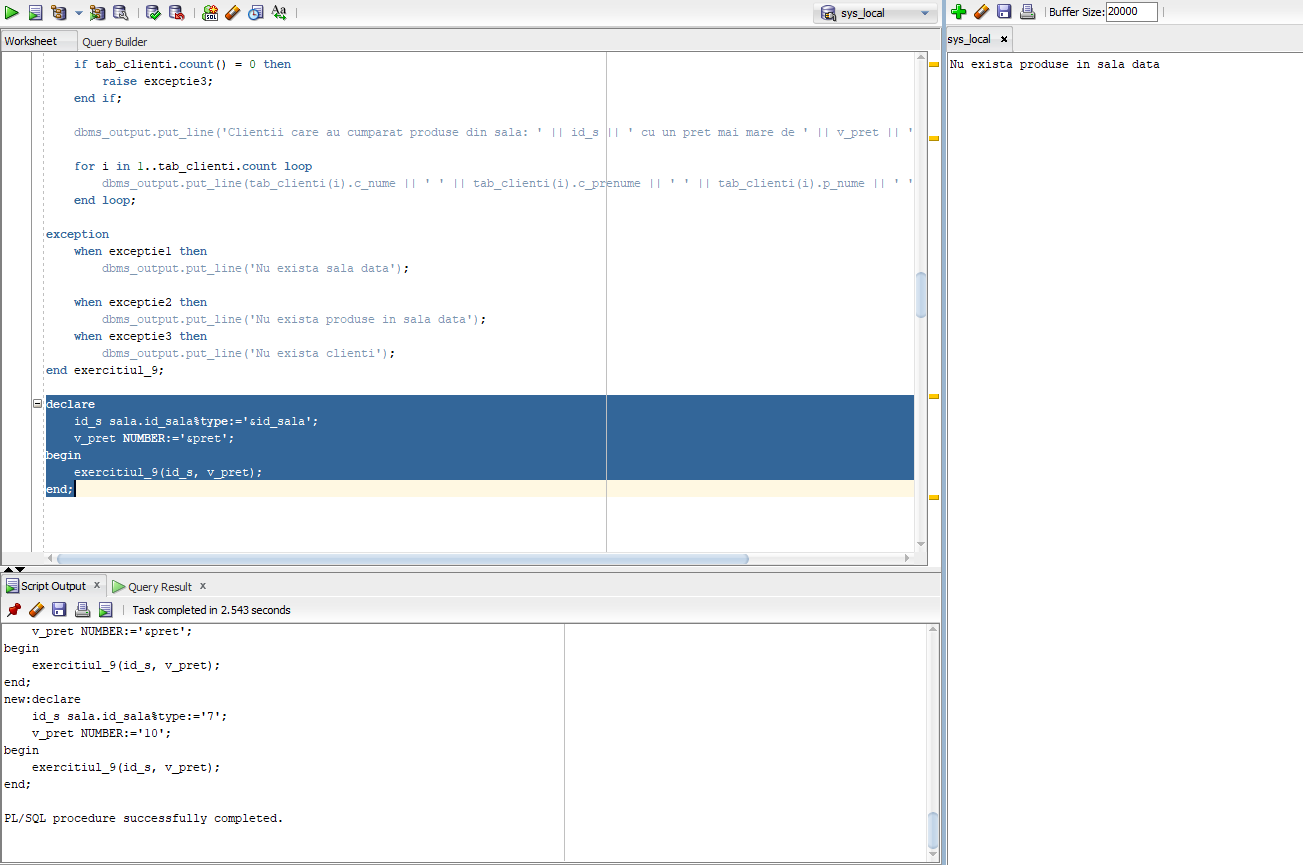
Id\_sala = 1 și pret = 10



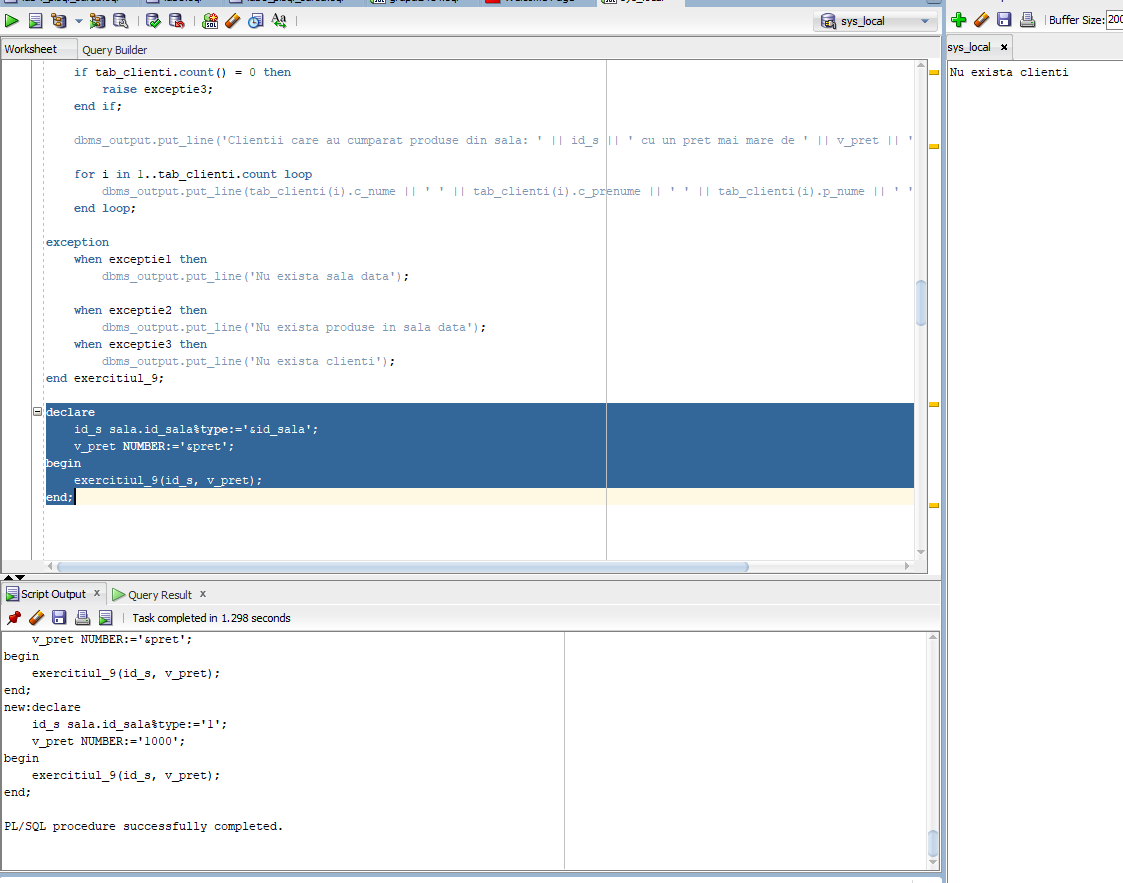
Id\_sala = 11 și pret = 0 (pentru excepția în care nu exista sala cu id\_ul dat)

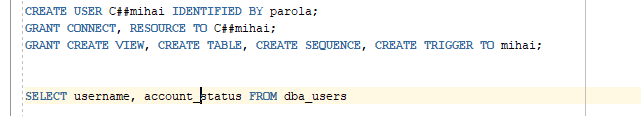


Id\_sala = 7 și pret = 10 (pentru a arata exceptia in care in sala nu exista produse)



Id\_sala = 1 și pret = 1000, pentru a arata excepția în care nu exista clienți:





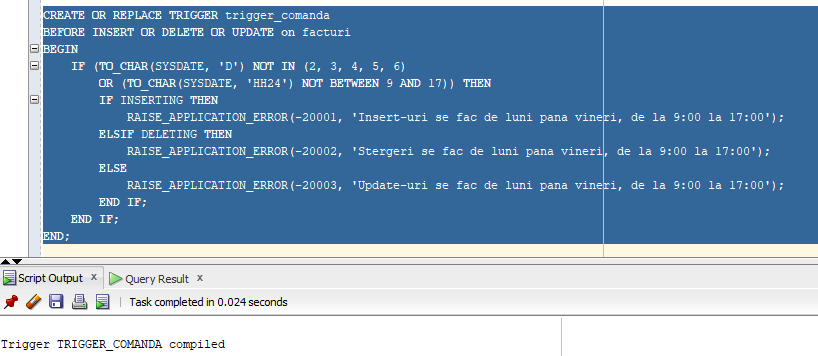
*10.Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.*

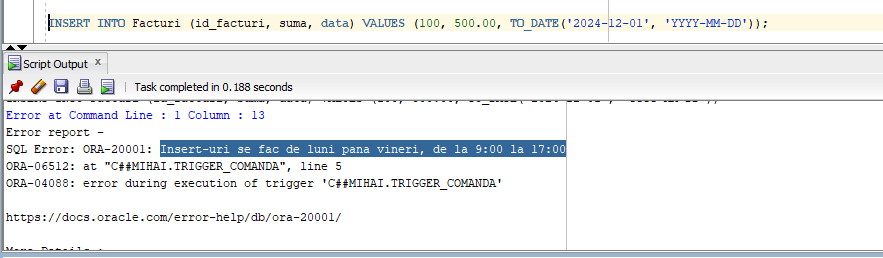
Îmi voi definii un trigger care sa nu permită modificarea integrității tabelului facturi (operatii de tip insert/update/delete) daca ziua nu este între Luni-Vineri, iar intervalul orar 9:00 - 17:00.

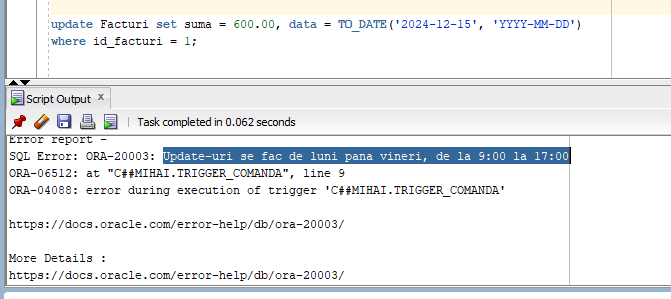
| CREATE OR REPLACE TRIGGER trigger\_comanda  BEFORE INSERT OR DELETE OR UPDATE on facturi  BEGIN  IF (TO\_CHAR(SYSDATE, 'D') NOT IN (2, 3, 4, 5, 6)  OR (TO\_CHAR(SYSDATE, 'HH24') NOT BETWEEN 9 AND 17)) THEN  IF INSERTING THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Insert-uri se fac de luni pana vineri, de la 9:00 la 17:00');  ELSIF DELETING THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Stergeri se fac de luni pana vineri, de la 9:00 la 17:00');  ELSE  RAISE\_APPLICATION\_ERROR(-20003, 'Update-uri se fac de luni pana vineri, de la 9:00 la 17:00');  END IF;  END IF;  END; |
| --- |

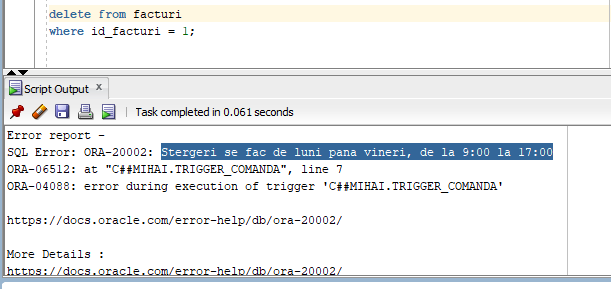
Am un IF care ia ziua, luni corespunde pentru 2, pana vineri la 6, și ora sa fie în intervalul 9 17.

Screenshot:









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

Îmi voi definii triggeri pentru a putea avea unicitate în cadrul facturilor, și anume:

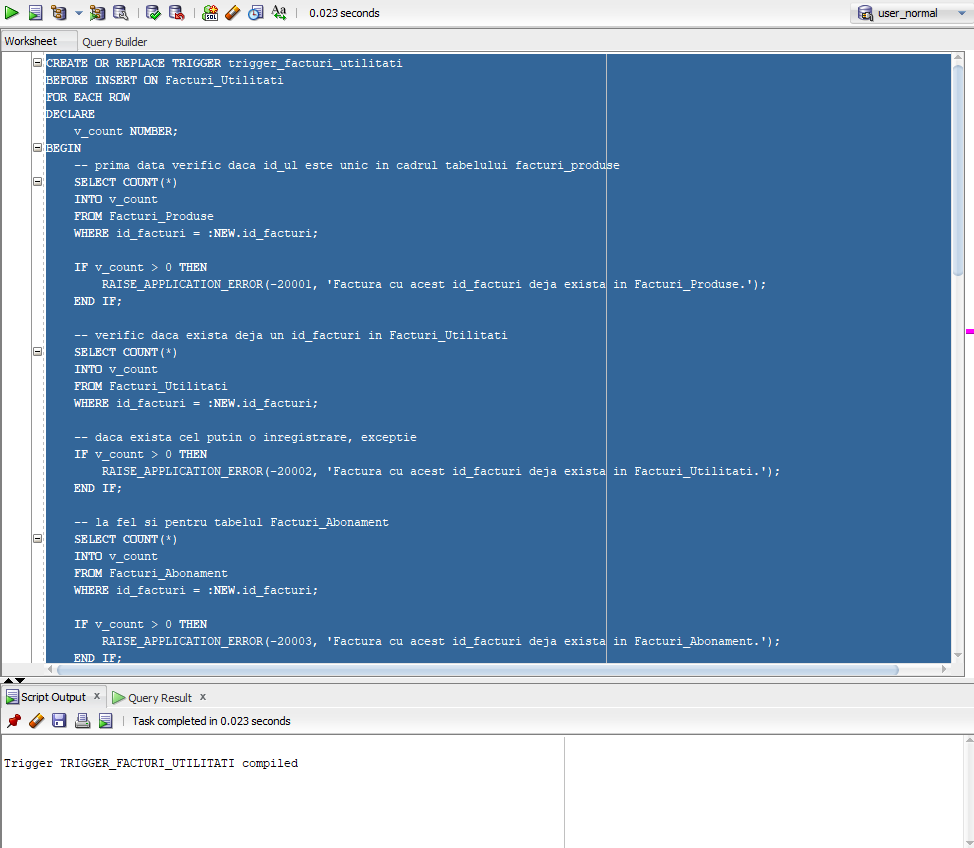
Daca exista deja un id\_factura in una din cele 3 categorii, sa nu mă lase sa inserez iar o pereche de tip (id\_factura, id\_factura\_tip) <- id\_factura sa fie același cu unu deja existent, iar id\_factura\_tip sa fie orice valoare.

Voi crea 3 triggere separate, unul pentru fiecare tip de factura care:

Verifica dacă exista sau nu acel ID de factura in cadrul celor 3 tabele, daca exista, dau un raise la eroare. (verificarea se va face cu un count).

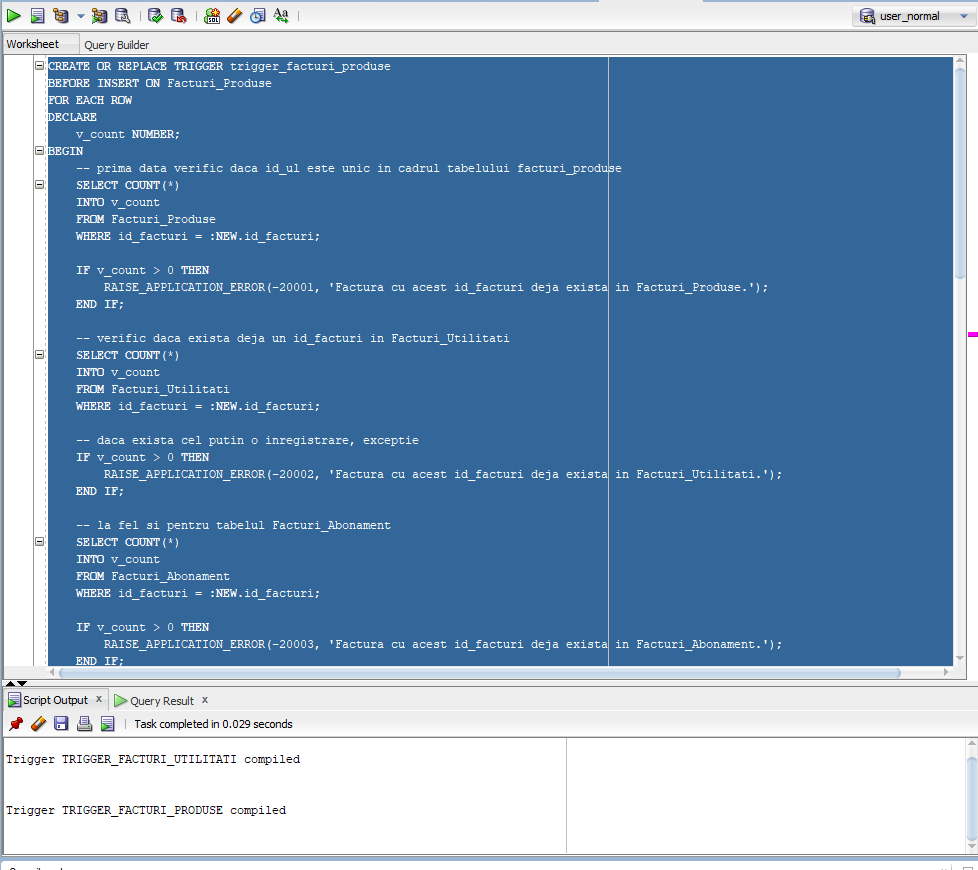
Pentru tabelul Facturi\_Utilitati:

| CREATE OR REPLACE TRIGGER trigger\_facturi\_utilitati  BEFORE INSERT ON Facturi\_Utilitati  FOR EACH ROW  DECLARE  v\_count NUMBER;  BEGIN  -- prima data verific daca id\_ul este unic in cadrul tabelului facturi\_produse  SELECT COUNT(\*)  INTO v\_count  FROM Facturi\_Produse  WHERE id\_facturi = :NEW.id\_facturi;    IF v\_count > 0 THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Factura cu acest id\_facturi deja exista in Facturi\_Produse.');  END IF;    -- verific daca exista deja un id\_facturi in Facturi\_Utilitati  SELECT COUNT(\*)  INTO v\_count  FROM Facturi\_Utilitati  WHERE id\_facturi = :NEW.id\_facturi;  -- daca exista cel putin o inregistrare, exceptie  IF v\_count > 0 THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Factura cu acest id\_facturi deja exista in Facturi\_Utilitati.');  END IF;  -- la fel si pentru tabelul Facturi\_Abonament  SELECT COUNT(\*)  INTO v\_count  FROM Facturi\_Abonament  WHERE id\_facturi = :NEW.id\_facturi;  IF v\_count > 0 THEN  RAISE\_APPLICATION\_ERROR(-20003, 'Factura cu acest id\_facturi deja exista in Facturi\_Abonament.');  END IF;  END; |
| --- |

Screenshot:  


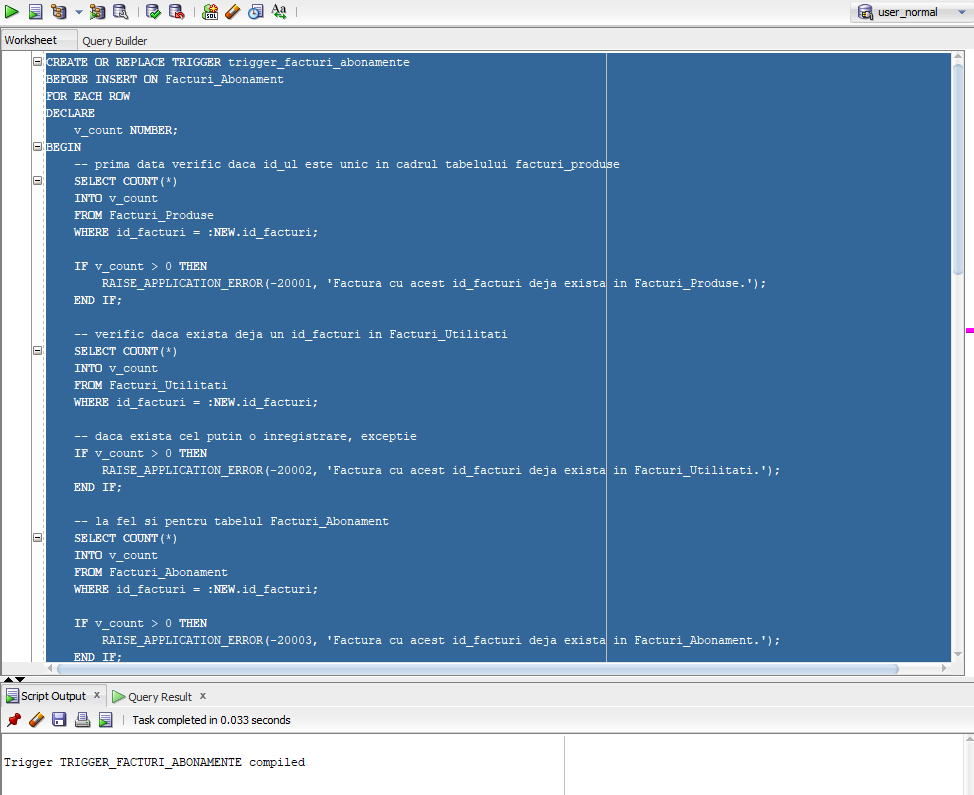
Pentru tabelul Facturi\_Produse:

| CREATE OR REPLACE TRIGGER trigger\_facturi\_produse  BEFORE INSERT ON Facturi\_Produse  FOR EACH ROW  DECLARE  v\_count NUMBER;  BEGIN  -- prima data verific daca id\_ul este unic in cadrul tabelului facturi\_produse  SELECT COUNT(\*)  INTO v\_count  FROM Facturi\_Produse  WHERE id\_facturi = :NEW.id\_facturi;    IF v\_count > 0 THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Factura cu acest id\_facturi deja exista in Facturi\_Produse.');  END IF;    -- verific daca exista deja un id\_facturi in Facturi\_Utilitati  SELECT COUNT(\*)  INTO v\_count  FROM Facturi\_Utilitati  WHERE id\_facturi = :NEW.id\_facturi;  -- daca exista cel putin o inregistrare, exceptie  IF v\_count > 0 THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Factura cu acest id\_facturi deja exista in Facturi\_Utilitati.');  END IF;  -- la fel si pentru tabelul Facturi\_Abonament  SELECT COUNT(\*)  INTO v\_count  FROM Facturi\_Abonament  WHERE id\_facturi = :NEW.id\_facturi;  IF v\_count > 0 THEN  RAISE\_APPLICATION\_ERROR(-20003, 'Factura cu acest id\_facturi deja exista in Facturi\_Abonament.');  END IF;  END; |
| --- |

Screenshot:  


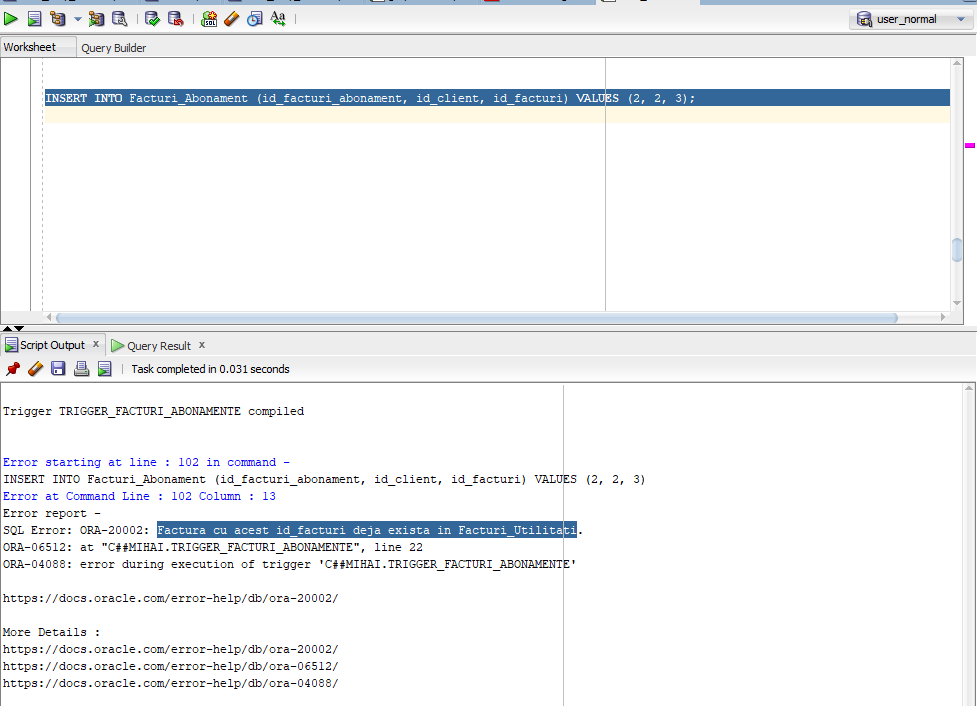
Pentru tabelul Facturi\_Abonamente:

| CREATE OR REPLACE TRIGGER trigger\_facturi\_abonamente  BEFORE INSERT ON Facturi\_Abonament  FOR EACH ROW  DECLARE  v\_count NUMBER;  BEGIN  -- prima data verific daca id\_ul este unic in cadrul tabelului facturi\_produse  SELECT COUNT(\*)  INTO v\_count  FROM Facturi\_Produse  WHERE id\_facturi = :NEW.id\_facturi;    IF v\_count > 0 THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Factura cu acest id\_facturi deja exista in Facturi\_Produse.');  END IF;    -- verific daca exista deja un id\_facturi in Facturi\_Utilitati  SELECT COUNT(\*)  INTO v\_count  FROM Facturi\_Utilitati  WHERE id\_facturi = :NEW.id\_facturi;  -- daca exista cel putin o inregistrare, exceptie  IF v\_count > 0 THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Factura cu acest id\_facturi deja exista in Facturi\_Utilitati.');  END IF;  -- la fel si pentru tabelul Facturi\_Abonament  SELECT COUNT(\*)  INTO v\_count  FROM Facturi\_Abonament  WHERE id\_facturi = :NEW.id\_facturi;  IF v\_count > 0 THEN  RAISE\_APPLICATION\_ERROR(-20003, 'Factura cu acest id\_facturi deja exista in Facturi\_Abonament.');  END IF;  END; |
| --- |

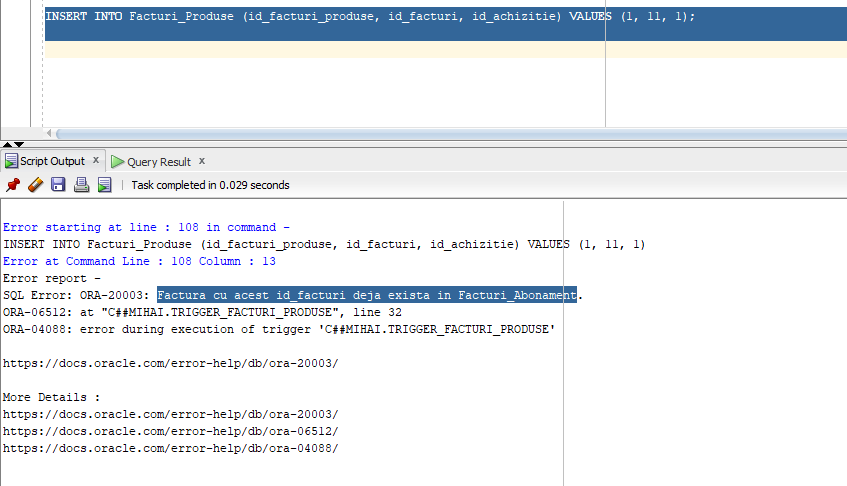


Vom testa triggeri în următorul mod:  
  
Pentru un id\_factura care este deja atribut unuia din cele 3 tipuri de facturi, încerc sa inserez o altă factură nouă:

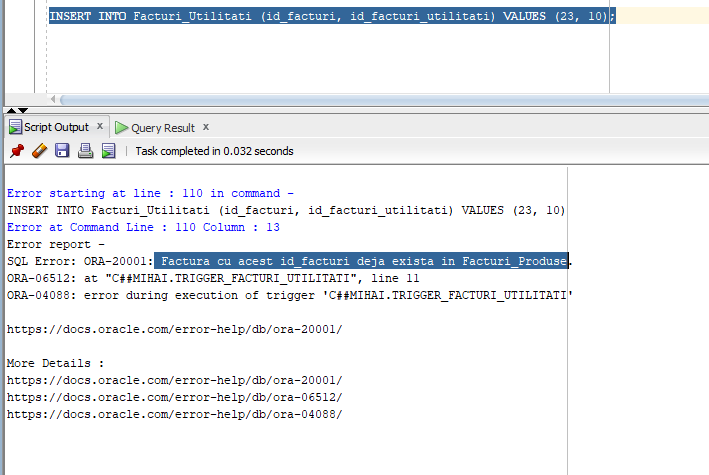
| INSERT INTO Facturi\_Abonament (id\_facturi\_abonament, id\_client, id\_facturi) VALUES (2, 2, 3); |
| --- |



| INSERT INTO Facturi\_Produse (id\_facturi\_produse, id\_facturi, id\_achizitie) VALUES (1, 11, 1); |
| --- |



| INSERT INTO Facturi\_Utilitati (id\_facturi, id\_facturi\_utilitati) VALUES (23, 10); |
| --- |



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

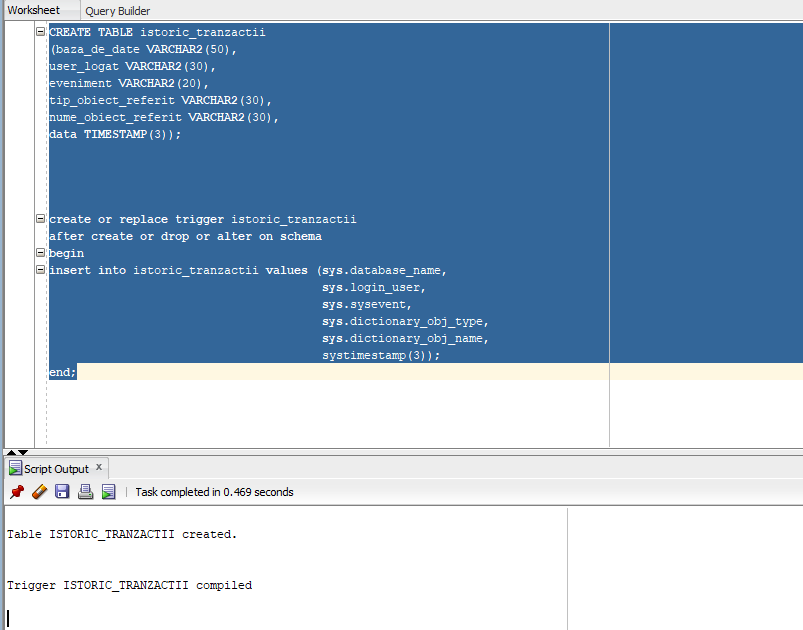
Vom crea un trigger care memorează fiecare tranzactie care are loc în cadrul schemei proiectului pentru a oferii un istoric al tuturor tranzacțiilor, ușor de accesat printr-un tabel numit ‘istoric\_tranzactii’.

Prima data vom creea tabelul:

| CREATE TABLE istoric\_tranzactii  (baza\_de\_date VARCHAR2(50),  user\_logat VARCHAR2(30),  eveniment VARCHAR2(20),  tip\_obiect\_referit VARCHAR2(30),  nume\_obiect\_referit VARCHAR2(30),  data TIMESTAMP(3)); |
| --- |

Trigger-ul in cauza:

| create or replace trigger istoric\_tranzactii  after create or drop or alter on schema  begin  insert into istoric\_tranzactii values (sys.database\_name,  sys.login\_user,  sys.sysevent,  sys.dictionary\_obj\_type,  sys.dictionary\_obj\_name,  systimestamp(3));  end; |
| --- |



Pentru a vedea ca totul functioneaza cum ne-am propus, am recompilat functia + procedura de la exercițiul 6 + am creat un nou tabel de test, caruia i-am dat alter, adăugându-i o nouă coloana, ulterior l-am sters, pentru a exemplifica toate cele 3 comenzi -> Create, Alter și Drop:

create table tabel\_de\_test ( id NUMBER PRIMARY KEY);

alter table tabel\_de\_test

add nume varchar2(10);

drop table tabel\_de\_test

