

**Proiect SGBD ORACLE**

***Gestiunea datelor pentru un salon de frumusețe***

Cuprins

[1. Obiectivul proiectului 4](#_Toc167000800)

[1.1. Descrierea bazei de date, relații dintre tabele 4](#_Toc167000801)

[2. Interacțiunea cu serverul Oracle prin intermediul comenzilor SQL (LMD/LDD) 5](#_Toc167000802)

[2.1. 5](#_Toc167000803)

[2.2. 6](#_Toc167000804)

[2.3. 7](#_Toc167000805)

[2.4. 8](#_Toc167000806)

[2.5. 9](#_Toc167000807)

[3. Structuri de control, cursori impliciți/expliciți 10](#_Toc167000808)

[3.1 10](#_Toc167000809)

[3.2 12](#_Toc167000810)

[3.3 14](#_Toc167000811)

[3.4 16](#_Toc167000812)

[3.5 18](#_Toc167000813)

[3.6 20](#_Toc167000814)

[4. Tratarea excepțiilor 22](#_Toc167000815)

[4.1 22](#_Toc167000816)

[4.2 24](#_Toc167000817)

[4.3 26](#_Toc167000818)

[4.4 28](#_Toc167000819)

[5. Proceduri, Funcții, Pachete 29](#_Toc167000820)

[5.1. PROCEDURI 29](#_Toc167000821)

[5.1.1. 29](#_Toc167000822)

[5.1.2. 32](#_Toc167000823)

[5.1.3. 34](#_Toc167000824)

[5.1.4. 36](#_Toc167000825)

[5.2. FUNCȚII 38](#_Toc167000826)

[5.2.1. 38](#_Toc167000827)

[5.2.2. 40](#_Toc167000828)

[5.2.3. 42](#_Toc167000829)

[5.2.4. 44](#_Toc167000830)

[5.2.5. 47](#_Toc167000831)

[5.3. PACHETE 49](#_Toc167000832)

[5.3.1 49](#_Toc167000833)

[5.3.2 52](#_Toc167000834)

[6. Triggeri 55](#_Toc167000835)

[6.1. 55](#_Toc167000836)

[6.2. 57](#_Toc167000837)

[6.3. 58](#_Toc167000838)

[6.4. 59](#_Toc167000839)

[7. Interfața cu utilizatorul 61](#_Toc167000840)

[7.1. Master Detail 61](#_Toc167000841)

[7.2. Set de pagini REPORT-FORM 63](#_Toc167000842)

[7.3. Set 2 de pagini REPORT-FORM 67](#_Toc167000843)

1. Obiectivul proiectului

## 1.1. Descrierea bazei de date, relații dintre tabele

Această bază de date ordonează și supraveghează datele unui salon de înfrumusețare numit Glam Beauty. Astfel, putem afla informații despre locația acestor saloane, despre serviciile care sunt prestate de anumiți angajați, clienții care beneficiază de acestea și ce fel de servicii au ales clienții. Putem prelucra în final aceste informații, având posibilitatea de a observa, modifica și calcula anumite date mai detaliat. Această bază de date verifică toate cele 3 tipuri de normalizare dintre care : atomicitate (nu există coloane sau grupuri de date repetate), dependență parțială (toate elementele, din fiecare tabelă, sunt dependente funcțional de cheia primară) și dependență tranzitivă (tabelele sunt în a doua formă normală și nu depind tranzitiv de cheia primară).

**Relațiile dintre tabele** sunt stabilite astfel:

Legătură **one to many** între ANGAJATI\_SALON și PROGRAMARI.

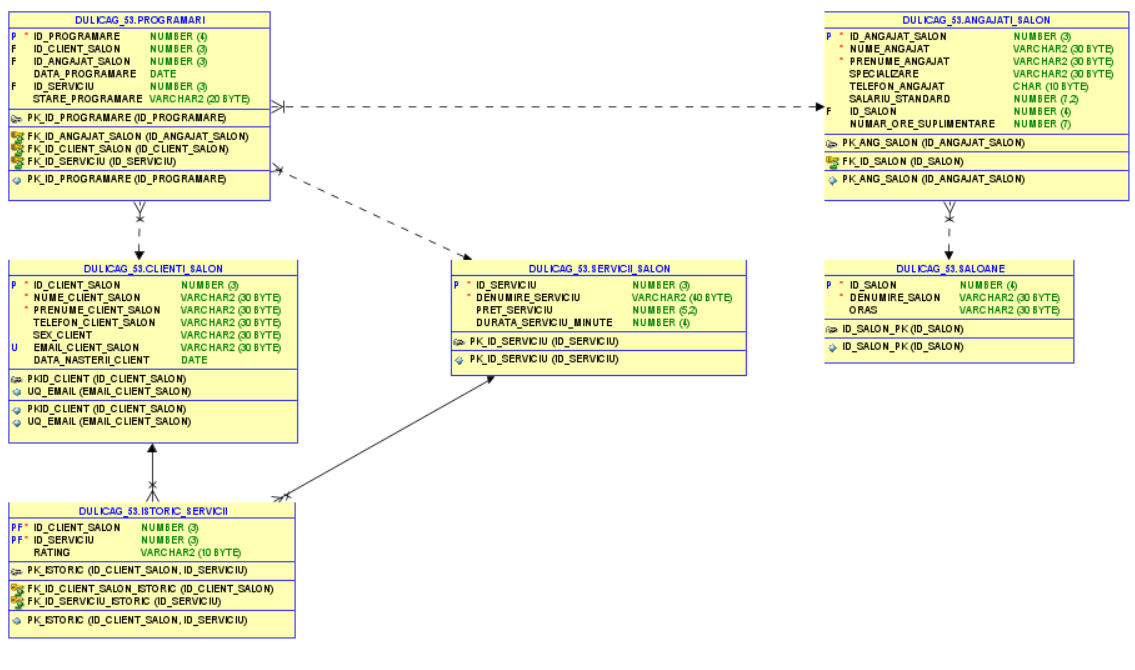
Legătură **one to many** între CLIENTI\_SALON și PROGRAMARI.

Legătură **one to many** între SALOANE și ANGAJATI\_SALON.

Legătură **one to many** între SERVICII\_SALON și PROGRAMARI.

Legătură **many to many** între CLIENTI\_SALON și SERVICII\_SALON, iar în acest caz am adăugat o tabelă suplimentară, ISTORIC\_SERVICII, care conține cheile primare ale tabelelor inițiale.

**Schema conceptuală**



# 2. Interacțiunea cu serverul Oracle prin intermediul comenzilor SQL (LMD/LDD)

## 2.1.

**Să se creeze o tabelă numită TABELA\_SUPLIMENTARA care are ca structură toate tabelele din ANGAJATI\_SALON și care conține toate înregistrările din aceasta.**

*BEGIN*

*EXECUTE IMMEDIATE 'CREATE TABLE TABELA\_SUPLIMENTARA AS SELECT \* FROM ANGAJATI\_SALON WHERE 1=1';*

*END;*

*/*

****

*SELECT \* FROM USER\_TABLES;*

**

*SELECT \* FROM TABELA\_SUPLIMENTARA*

**REZULTAT:**

****

## 2.2.

**Să se adauge în tabela nou creată coloana comision. Să se introducă o valoarea a comisionului de 0.3 pentru angajații care au salariul mai mic de 5000 lei și 0.2 pentru angajații care au un salariu mai mare de 5000 lei.**

*DECLARE*

*V\_SIR VARCHAR2(200);*

*BEGIN*

*V\_SIR := 'ALTER TABLE TABELA\_SUPLIMENTARA ADD (COMISION NUMBER(9,2))';*

*EXECUTE IMMEDIATE V\_SIR;*

*DBMS\_OUTPUT.PUT\_LINE('Coloana COMISION a fost adăugata');*

*END;*

*/*

*BEGIN*

*UPDATE TABELA\_SUPLIMENTARA*

*SET COMISION = CASE*

*WHEN SALARIU\_STANDARD < 5000 THEN 0.3*

*ELSE 0.2*

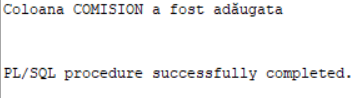
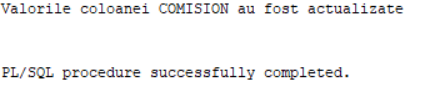
*END*

*WHERE ID\_ANGAJAT\_SALON=&ID;*

*DBMS\_OUTPUT.PUT\_LINE('Valorile coloanei COMISION au fost actualizate');*

*END;*

*/*

** **

*SELECT ID\_ANGAJAT\_SALON,NUME\_ANGAJAT,COMISION FROM TABELA\_SUPLIMENTARA WHERE ID\_ANGAJAT\_SALON=10;*

**REZULTAT: **

## 2.3.

**Să se adauge în tabela TABELA\_SUPLIMENTARA o înregistrare noua (cu id=1). Ulterior acestui angajat i se va modifica id-ul cu o valoare egală cu maximul dintre id-urile celorlalți angajați la care se adaugă valoarea 1 .**

*BEGIN*

*INSERT INTO TABELA\_SUPLIMENTARA VALUES (1,'Deaconeasa','Adina','Make-up',0789675342,4700,1,2,10,0.3);*

*END;*

*SELECT ID\_ANGAJAT\_SALON,NUME\_ANGAJAT,PRENUME\_ANGAJAT,SPECIALIZARE,COMISION FROM TABELA\_SUPLIMENTARA WHERE ID\_ANGAJAT\_SALON=1;*

**

*DECLARE*

*V\_ID NUMBER;*

*BEGIN*

*SELECT MAX(ID\_ANGAJAT\_SALON)INTO V\_ID FROM TABELA\_SUPLIMENTARA;*

*UPDATE TABELA\_SUPLIMENTARA SET ID\_ANGAJAT\_SALON=V\_ID+1 WHERE ID\_ANGAJAT\_SALON=1;*

*END;*

**REZULTAT:**

****

****

## 2.4.

**Să se șteargă înregistrarea care conține cea mai mica valoarea a comisionului și să se afișeze înainte de ștergere informații despre angajatul care are acest comision.**

*DECLARE*

*V\_ID\_ANGAJAT TABELA\_SUPLIMENTARA.ID\_ANGAJAT\_SALON%TYPE;*

*V\_NUME\_ANGAJAT TABELA\_SUPLIMENTARA.NUME\_ANGAJAT%TYPE;*

*V\_COMISION\_MIN NUMBER(9,2);*

*BEGIN*

*SELECT MIN(COMISION) INTO V\_COMISION\_MIN FROM TABELA\_SUPLIMENTARA;*

*SELECT ID\_ANGAJAT\_SALON, NUME\_ANGAJAT INTO V\_ID\_ANGAJAT, V\_NUME\_ANGAJAT*

*FROM TABELA\_SUPLIMENTARA*

*WHERE COMISION = V\_COMISION\_MIN;*

*DBMS\_OUTPUT.PUT\_LINE('ID ANGAJAT: ' || V\_ID\_ANGAJAT || ' NUME ANGAJAT: ' || V\_NUME\_ANGAJAT || ' COMISION: ' || V\_COMISION\_MIN);*

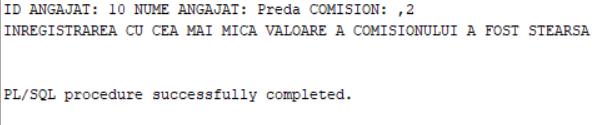
*DELETE FROM TABELA\_SUPLIMENTARA WHERE COMISION = V\_COMISION\_MIN;*

*DBMS\_OUTPUT.PUT\_LINE('INREGISTRAREA CU CEA MAI MICA VALOARE A COMISIONULUI A FOST STEARSA');*

*END;*

*/*

**REZULTAT:**

****

## 2.5.

**Să se șteargă tabela TABELA\_SUPLIMENTARA. Verificați dacă tabela încă există în baza de date.**

*DECLARE*

*V\_COUNT NUMBER;*

*BEGIN*

*SELECT COUNT(\*) INTO V\_COUNT FROM USER\_TABLES WHERE TABLE\_NAME = 'TABELA\_SUPLIMENTARA';*

*IF V\_COUNT > 0 THEN*

*EXECUTE IMMEDIATE 'DROP TABLE TABELA\_SUPLIMENTARA';*

*DBMS\_OUTPUT.PUT\_LINE('TABELA TABELA\_SUPLIMENTARA A FOST STEARSA');*

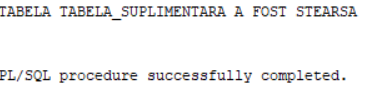
*ELSE*

*DBMS\_OUTPUT.PUT\_LINE('TABELA TABELA\_SUPLIMENTARA NU EXISTA.');*

*END IF;*

*END;*

*/*

****

3. Structuri de control, cursori impliciți/expliciți

## 3.1

**Să se mărească cu 10 lei prețul primelor 2 servicii din salon, care au fost cel mai des solicitate. Să se afișeze, printr-o variabilă de mediu, numărul rândurilor afectate de comanda UPDATE și să se afiseze de asemenea prețurile înainte și dupa modificare. Să se arate ce se va întâmpla în urma utilizării comenzii ROLLBACK.**

* *Cursor explicit FOR LOOP, cursor implicit*

*VARIABLE g\_nr\_servicii VARCHAR2(30)*

*DECLARE*

*CURSOR servicii\_cursor IS SELECT s.id\_serviciu, s.denumire\_serviciu,s.pret\_serviciu, COUNT(i.id\_client\_salon) nr\_clienti*

*FROM servicii\_salon s*

*JOIN istoric\_servicii i*

*ON i.id\_serviciu=s.id\_serviciu*

*GROUP BY s.id\_serviciu, s.denumire\_serviciu,s.pret\_serviciu*

*ORDER BY nr\_clienti DESC;*

*v\_denumire servicii\_salon.denumire\_serviciu%TYPE;*

*v\_pret servicii\_salon.pret\_serviciu%TYPE;*

*v\_nr\_actualizari NUMBER := 0;*

*BEGIN*

*FOR servicii\_rec IN servicii\_cursor LOOP*

*DBMS\_OUTPUT.PUT\_LINE(servicii\_rec.id\_serviciu||'DENUMIRE: '||servicii\_rec.denumire\_serviciu||' -PRET: '||servicii\_rec.pret\_serviciu||' -SOLICITAT DE: '||servicii\_rec.nr\_clienti);*

*UPDATE servicii\_salon SET pret\_serviciu=pret\_serviciu+10*

*WHERE id\_serviciu=servicii\_rec.id\_serviciu;*

*SELECT denumire\_serviciu,pret\_serviciu INTO v\_denumire,v\_pret FROM servicii\_salon WHERE id\_serviciu=servicii\_rec.id\_serviciu;*

*DBMS\_OUTPUT.PUT\_LINE('DUPA UPDATE: '||v\_denumire||' -PRET NOU:'||v\_pret);*

*v\_nr\_actualizari := v\_nr\_actualizari + SQL%ROWCOUNT;*

*EXIT WHEN servicii\_cursor%ROWCOUNT = 2;*

*END LOOP;*

*:g\_nr\_servicii:=to\_char('NUMAR RANDURI MODIFICATE: '||v\_nr\_actualizari);*

*ROLLBACK;*

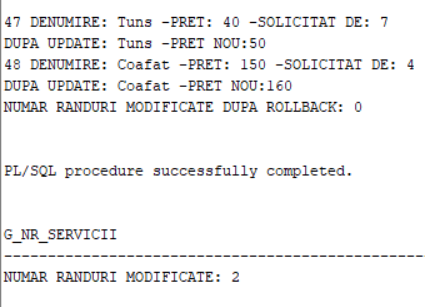
*DBMS\_OUTPUT.PUT\_LINE('NUMAR RANDURI MODIFICATE DUPA ROLLBACK: '||SQL%ROWCOUNT);*

*END;*

*/*

*PRINT g\_nr\_servicii*

**REZULTAT**



## 3.2

**Să se afișeze programările care au fost anulate în anul 2023 între luna ianuarie si mai. Starea programării va fi introdusă de la tastatură și la final afișați id-ul, data, starea programării. Tratați situația în care cursorul a fost deja deschis printr-o excepție.**

* *Cursor explicit OPEN-LOOP-FECTH-CLOSE, structura repetitivă LOOP*

*ACCEPT G\_STARE\_PROGRAMARE PROMPT 'INTRODUCETI STAREA PROGRAMARII: '*

*DECLARE*

*CURSOR PROGRAMARI\_CURSOR IS SELECT ID\_PROGRAMARE,DATA\_PROGRAMARE,STARE\_PROGRAMARE FROM PROGRAMARI*

*WHERE DATA\_PROGRAMARE BETWEEN TO\_DATE('01.01.2023','DD.MM.YYYY') AND TO\_DATE('01.05.2023','DD.MM.YYYY')*

*AND STARE\_PROGRAMARE='&G\_STARE\_PROGRAMARE'*

*ORDER BY ID\_PROGRAMARE;*

*PROGRAMARI\_REC PROGRAMARI\_CURSOR%ROWTYPE;*

*BEGIN*

*IF PROGRAMARI\_CURSOR%ISOPEN=FALSE THEN*

*OPEN PROGRAMARI\_CURSOR;*

*END IF;*

*LOOP*

*FETCH PROGRAMARI\_CURSOR INTO PROGRAMARI\_REC;*

*EXIT WHEN PROGRAMARI\_CURSOR%NOTFOUND;*

*DBMS\_OUTPUT.PUT\_LINE(PROGRAMARI\_REC.ID\_PROGRAMARE||'-DATA:'||PROGRAMARI\_REC.DATA\_PROGRAMARE||'-STARE PROGRAMARE:'||PROGRAMARI\_REC.STARE\_PROGRAMARE);*

*END LOOP;*

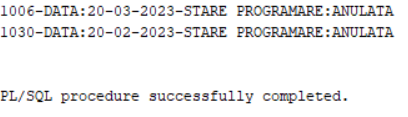
*CLOSE PROGRAMARI\_CURSOR;*

*EXCEPTION*

*WHEN CURSOR\_ALREADY\_OPEN THEN DBMS\_OUTPUT.PUT\_LINE('CURSOR DESCHIS !'); -- Excepție pre-definită*

*END;*

**REZULTAT**



## 3.3

**Să se afișeze clienții care au avut mai mult de o programare, să se afișeze date despre client, și despre serviciul pe care l-a solicitat.**

* *Cursor explicit (INLINE) FOR LOOP*

*DECLARE*

*CURSOR clienti\_cursor IS SELECT c.id\_client\_salon, c.nume\_client\_salon,COUNT(i.id\_serviciu) FROM clienti\_salon c*

*JOIN istoric\_servicii i ON c.id\_client\_salon=i.id\_client\_salon*

*GROUP BY c.id\_client\_salon, c.nume\_client\_salon*

*HAVING COUNT(i.id\_serviciu)>1;*

*CURSOR programari\_cursor(id clienti\_salon.id\_client\_salon%TYPE) IS SELECT p.id\_programare,s.denumire\_serviciu FROM programari p*

*JOIN servicii\_salon s ON p.id\_serviciu=s.id\_serviciu*

*WHERE p.id\_client\_salon=id*

*GROUP BY p.id\_programare,s.denumire\_serviciu;*

*BEGIN*

*FOR clienti\_rec IN clienti\_cursor LOOP*

*DBMS\_OUTPUT.PUT\_LINE(clienti\_rec.id\_client\_salon||'-NUME:'||clienti\_rec.nume\_client\_salon);*

*FOR programari\_rec IN programari\_cursor(clienti\_rec.id\_client\_salon) LOOP*

*DBMS\_OUTPUT.PUT\_LINE(programari\_rec.id\_programare||'- '||programari\_rec.denumire\_serviciu);*

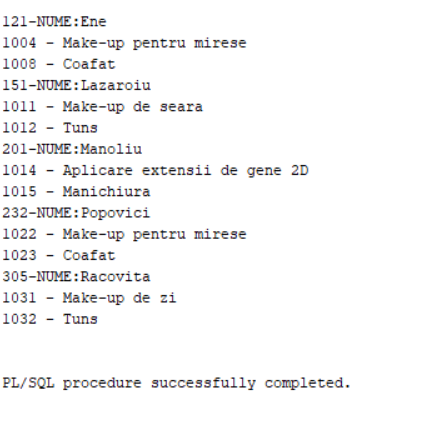
*END LOOP;*

*END LOOP;*

*END;*

*/*

**REZULTAT**



## 3.4

**Afișează angajații care au ca specializare Manichiura/Pedichiura și orașul în care lucrează. Pentru găsirea angajaților nu se va folosi un cursor, ci parcurgerea se va face cu ajutorul structurilor de control. ( ! id-urile angajaților nu sunt numere consecutive) . Datele din tabele se vor prelua cu ajutorul tipului de variabilă RECORD, ROWTYPE.**

* *Structură de control repetitivă FOR, structură alternativă IF*

*DECLARE*

*ang\_rec angajati\_salon%ROWTYPE;*

*salon\_rec saloane%ROWTYPE;*

*v\_max NUMBER;*

*v\_min NUMBER;*

*v\_verificare NUMBER;*

*BEGIN*

*SELECT MAX(id\_angajat\_salon), MIN(id\_angajat\_salon) INTO v\_max, v\_min FROM ANGAJATI\_SALON;*

*FOR ang\_rec\_id IN v\_min..v\_max LOOP*

*SELECT COUNT(id\_angajat\_salon) INTO v\_verificare FROM angajati\_salon WHERE id\_angajat\_salon = ang\_rec\_id;*

*IF v\_verificare = 1 THEN*

*SELECT nume\_angajat, specializare INTO ang\_rec.nume\_angajat, ang\_rec.specializare FROM ANGAJATI\_SALON WHERE id\_angajat\_salon = ang\_rec\_id;*

*SELECT s.oras INTO salon\_rec.oras FROM saloane s JOIN angajati\_salon a ON s.id\_salon = a.id\_salon WHERE a.id\_angajat\_salon = ang\_rec\_id;*

*IF ang\_rec.specializare = 'Manichiura/Pedichiura' THEN*

*DBMS\_OUTPUT.PUT\_LINE(ang\_rec.nume\_angajat || ' - ' || ang\_rec.specializare || ' - ' || salon\_rec.oras);*

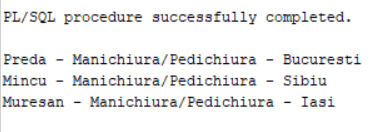
*END IF;*

*END IF;*

*END LOOP;*

*END;*

**REZULTAT**



## 3.5

**Să se aplice o reducere de 10% pentru clienții care au beneficiat de servicii cu o valoare mai mare de 300 de lei (se ține cont de faptul că un client putea să solicite mai multe servicii). Se va ultiliza un tip de date RECORD pentru a prelua datele din tabele și la final se vor afișa date în legătură cu reducerea obținută, împreună cu numele clientului care a beneficiat de reducere.**

* *Structură de control repetitivă WHILE, structură alternativă IF*

*DECLARE*

*TYPE tip\_record IS RECORD (*

*v\_suma\_totala NUMBER,*

*v\_id\_client\_salon clienti\_salon.id\_client\_salon%TYPE,*

*v\_nume\_client clienti\_salon.nume\_client\_salon%TYPE,*

*v\_prenume\_client clienti\_salon.prenume\_client\_salon%TYPE,*

*v\_reducere\_totala NUMBER := 0);*

*client\_rec tip\_record;*

*v\_max NUMBER;*

*v\_min NUMBER;*

*v\_verificare NUMBER;*

*BEGIN*

*SELECT MAX(id\_client\_salon), MIN(id\_client\_salon) INTO v\_max, v\_min FROM clienti\_salon;*

*WHILE v\_min<=v\_max LOOP*

*SELECT COUNT(id\_client\_salon) INTO v\_verificare FROM clienti\_salon WHERE id\_client\_salon = v\_min;*

*IF v\_verificare = 1 THEN*

*SELECT SUM(S.PRET\_SERVICIU) AS SUMA\_TOTALA, I.ID\_CLIENT\_SALON*

*INTO client\_rec.v\_suma\_totala, client\_rec.v\_id\_client\_salon*

*FROM SERVICII\_SALON S*

*JOIN ISTORIC\_SERVICII I ON S.ID\_SERVICIU = I.ID\_SERVICIU*

*WHERE I.ID\_CLIENT\_SALON = v\_min*

*GROUP BY I.ID\_CLIENT\_SALON;*

*SELECT nume\_client\_salon, prenume\_client\_salon*

*INTO client\_rec.v\_nume\_client, client\_rec.v\_prenume\_client*

*FROM clienti\_salon*

*WHERE id\_client\_salon = v\_min;*

*IF client\_rec.v\_suma\_totala > 300 THEN*

*client\_rec.v\_reducere\_totala := client\_rec.v\_suma\_totala \* 0.1;*

*DBMS\_OUTPUT.PUT\_LINE('Client: ' || client\_rec.v\_nume\_client || ' ' || client\_rec.v\_prenume\_client || ', Reducere aplicată: ' || client\_rec.v\_reducere\_totala);*

*END IF;*

*END IF;*

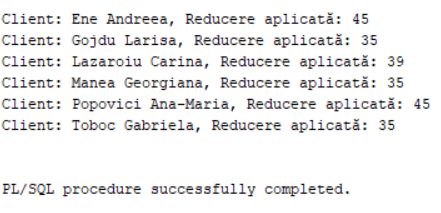
*v\_min := v\_min + 1;*

*END LOOP;*

*END;*

*/*

**REZULTAT**



## 3.6

**Implementați un bloc PL/SQL, astfel încât să se mărească prețul serviciului disponibil în salon, al cărui id este dat de la tastatura, în funcție de durata petrecută de angajat pentru a executa acest serviciu astfel:**

* **Dacă angajatul petrece între 60 și 99 de minute pentru a face acest serviciu, prețul se va mări cu valoarea 5**
* **Dacă angajatul petrece între 100 și 150 de minute pentru a face acest serviciu, prețul se va mări cu valoarea 10**
* **Altfel, nu se va schimba prețul**

**Să se afiseze informații despre acest serviciu cu datele inițiale și datele dupa actualizare.**

* *Structură alternativă CASE EXPRESSIONS, structură alternativă IF, cursor implicit*

*ACCEPT G\_ID\_SERVICIU PROMPT 'INTRODUCETI ID-UL SERVICIULUI DISPONIBIL IN SALON:'*

*DECLARE*

*V\_REC SERVICII\_SALON%ROWTYPE;*

*V\_NOU NUMBER;*

*V\_VECHI NUMBER;*

*EXCEPTIE EXCEPTION;*

*BEGIN*

*SELECT \* INTO V\_REC FROM SERVICII\_SALON WHERE ID\_SERVICIU=&G\_ID\_SERVICIU;*

*DBMS\_OUTPUT.PUT\_LINE(V\_REC.ID\_SERVICIU||' DENUMIRE:'||V\_REC.DENUMIRE\_SERVICIU||' PRET INITIAL:'||V\_REC.PRET\_SERVICIU||' DURATA:'||V\_REC.DURATA\_SERVICIU\_MINUTE);*

*V\_VECHI:=V\_REC.PRET\_SERVICIU;*

*V\_REC.PRET\_SERVICIU:=*

*CASE*

*WHEN V\_REC.DURATA\_SERVICIU\_MINUTE BETWEEN 60 AND 99 THEN V\_REC.PRET\_SERVICIU+5*

*WHEN V\_REC.DURATA\_SERVICIU\_MINUTE BETWEEN 100 AND 150 THEN V\_REC.PRET\_SERVICIU+10*

*ELSE V\_REC.PRET\_SERVICIU*

*END;*

*UPDATE SERVICII\_SALON SET PRET\_SERVICIU = V\_REC.PRET\_SERVICIU WHERE ID\_SERVICIU = V\_REC.ID\_SERVICIU;*

*IF V\_VECHI != V\_REC.PRET\_SERVICIU THEN*

*IF SQL%ROWCOUNT > 0 THEN*

*SELECT PRET\_SERVICIU INTO V\_NOU FROM SERVICII\_SALON WHERE ID\_SERVICIU=V\_REC.ID\_SERVICIU;*

*DBMS\_OUTPUT.PUT\_LINE('ACTUALIZARE REUSITA: PRET NOU:'||V\_NOU);*

*END IF;*

*ELSE*

*RAISE EXCEPTIE;*

*END IF;*

*EXCEPTION*

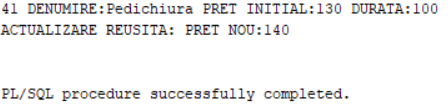
*WHEN EXCEPTIE THEN DBMS\_OUTPUT.PUT\_LINE('DURATA PENTRU ACEST SERVICIU ESTE PREA MICA, NU SE VA ACTUALIZA PRETUL');*

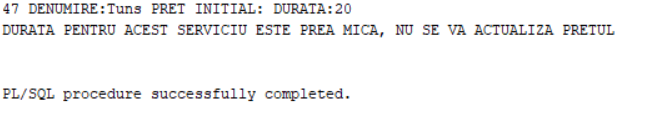
*WHEN NO\_DATA\_FOUND THEN DBMS\_OUTPUT.PUT\_LINE('NU AVEM PRODUSE CU ACEST ID');*

*END;*

*/*

**REZULTAT**





# 4. Tratarea excepțiilor

## 4.1

**Selectați toți angajații care au ca specialitate Manichiura/Pedichiura. Afișați informații despre aceștia referitoare la nume, prenume și salonul în care lucrează. Dacă există mai mulți angajați care dețin această specialitate, tratați excepția corespunzător, astfel încât să aveți posibilitatea de a vedea acei angajați.**

* *Excepție pre-definită*

*ACCEPT g\_specializare PROMPT 'Introduceti specializarea: '*

*DECLARE*

*v\_nume\_angajat angajati\_salon.nume\_angajat%TYPE;*

*v\_prenume\_angajat angajati\_salon.prenume\_angajat%TYPE;*

*v\_specializare angajati\_salon.specializare%TYPE:='&g\_specializare';*

*v\_id\_salon angajati\_salon.id\_salon%TYPE;*

*v\_oras saloane.oras%TYPE;*

*BEGIN*

*SELECT a.nume\_angajat,a.prenume\_angajat,a.id\_salon,s.oras*

*INTO v\_nume\_angajat,v\_prenume\_angajat,v\_id\_salon,v\_oras*

*FROM angajati\_salon a*

*JOIN saloane s ON a.id\_salon=s.id\_salon*

*WHERE a.specializare=v\_specializare;*

*EXCEPTION*

*WHEN TOO\_MANY\_ROWS THEN DBMS\_OUTPUT.PUT\_LINE('Prea multe randuri returnate de select');*

*DECLARE*

*CURSOR ang\_cursor IS SELECT a.nume\_angajat,a.prenume\_angajat,a.id\_salon,s.oras*

*FROM angajati\_salon a*

*JOIN saloane s ON a.id\_salon=s.id\_salon*

*WHERE a.specializare=v\_specializare;*

*BEGIN*

*FOR ang\_rec IN ang\_cursor LOOP*

*DBMS\_OUTPUT.PUT\_LINE('Angajatul '||ang\_rec.nume\_angajat||' '||ang\_rec.prenume\_angajat||' lucreaza in '||ang\_rec.id\_salon||'-'||ang\_rec.oras);*

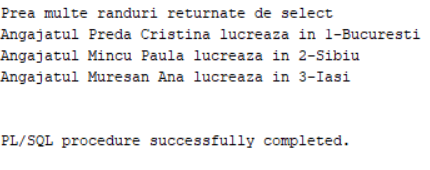
*END LOOP;*

*END;*

*END;*

*/*

**REZULTAT**



## 4.2

**Introduceți o programare noua (data\_programare, id\_serviciu). Dacă în data respectiva, are loc o programare asemănătoare, invocați o excepția care spune că acea dată este ocupată. Dacă acea dată nu este ocupată, confirmați stabilirea programării.**

**Exemplu:**

* **în data de 02-02-2024, avem de prestat un serviciu cu id 40 – ocupat dacă**

**se dorește serviciul cu id=40, liber pentru celelalte.**

* **în data de 03-04-2024 – liber pentru orice serviciu.**
* *Excepție definită de utilizator*

*ACCEPT g\_data\_programare PROMPT 'Introduceti data programarii:'*

*ACCEPT g\_id\_serviciu PROMPT 'Introduceti id-ul serviciului pe care il doriti:'*

*DECLARE*

*exceptie EXCEPTION;*

*v\_data\_noua programari.data\_programare%TYPE := TO\_DATE('&g\_data\_programare', 'DD-MM-YYYY');*

*v\_id\_serviciu\_nou programari.id\_serviciu%TYPE := &g\_id\_serviciu;*

*CURSOR prog\_cursor IS SELECT id\_programare, data\_programare, id\_serviciu FROM programari;*

*gasit NUMBER:=0;*

*BEGIN*

*FOR prog\_rec IN prog\_cursor LOOP*

*IF prog\_rec.data\_programare = v\_data\_noua AND prog\_rec.id\_serviciu = v\_id\_serviciu\_nou THEN*

*RAISE exceptie;*

*ELSE*

*DBMS\_OUTPUT.PUT\_LINE('Programare stabilita in data de ' || v\_data\_noua);*

*gasit:=1;*

*END IF;*

*EXIT WHEN gasit=1;*

*END LOOP;*

*EXCEPTION*

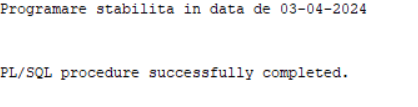
*WHEN exceptie THEN*

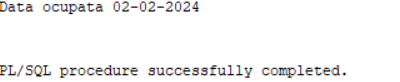
*DBMS\_OUTPUT.PUT\_LINE('Data ocupata');*

*END;*

*/*

**REZULTAT**





## 4.3

**Să se modifice programarea unui client al cărui id îl introduc de la tastatură, astfel încât să aibă id\_serviciu egal cu 41. Dacă programarea acestui client a fost anulată sau finalizată, se va reflecta acest aspect printr-o excepție, iar id\_serviciu va rămâne la valoarea inițială. Dacă clientul nu este înregistrat în baza de date, se va invoca altă excepție, cu aceleași condiții.**

* *Excepție definită de utilizator*

*ACCEPT g\_id\_client PROMPT 'Introduceti id-ul clientului:'*

*DECLARE*

*exceptie1 EXCEPTION;*

*PRAGMA EXCEPTION\_INIT(exceptie1, -20999);*

*v\_stare\_programare programari.stare\_programare%TYPE;*

*exceptie2 EXCEPTION;*

*v\_id programari.id\_client\_salon%TYPE := &g\_id\_client;*

*BEGIN*

*UPDATE programari SET ID\_SERVICIU = 41 WHERE id\_client\_salon = &g\_id\_client;*

*IF SQL%ROWCOUNT = 0 THEN --Cursor implicit*

*RAISE\_APPLICATION\_ERROR(-20999, 'Clientul acesta nu este inregistrat');*

*ELSE*

*SELECT stare\_programare INTO v\_stare\_programare FROM programari WHERE id\_client\_salon = &g\_id\_client;*

*IF v\_stare\_programare IN ('ANULATA', 'FINALIZATA') THEN*

*RAISE exceptie2;*

*ELSE*

*DBMS\_OUTPUT.PUT\_LINE('Clientului cu id: ' || v\_id || ' i s-a modificat programarea cu succes');*

*END IF;*

*END IF;*

*EXCEPTION*

*WHEN exceptie1 THEN*

*DBMS\_OUTPUT.PUT\_LINE('MESAJ: ' || SQLERRM || ' COD: ' || SQLCODE);*

*ROLLBACK;*

*WHEN exceptie2 THEN*

*DBMS\_OUTPUT.PUT\_LINE('Programare finalizata sau anulata');*

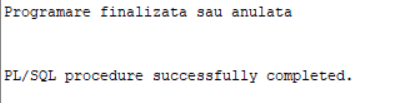
*ROLLBACK;*

*END;*

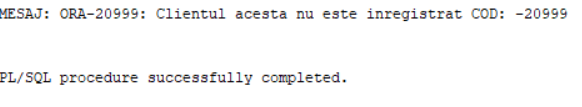
*/*

**REZULTAT**

* **Clientul cu id 111, are starea programării ANULATĂ.**



* **Clientul cu id 111, nu există.**



* **Clientul cu id 101, există.**



## 4.4

**Inserați un nou salon în tabela SALOANE, cu specificația să introduceți doar id\_salon. Folosiți codul de eroare -1400 știind ca denumire\_salon din tabela saloane are o restricție NOT NULL. Să se evidențieze numele utilizatorului, data, codul și mesajul de eroare.**

* *Excepție non-predefinită*

*DECLARE*

*exceptie EXCEPTION;*

*PRAGMA EXCEPTION\_INIT(exceptie,-1400);*

*v\_id angajati\_salon.id\_angajat\_salon%TYPE;*

*BEGIN*

*INSERT INTO saloane (id\_salon) VALUES (4);*

*DBMS\_OUTPUT.PUT\_LINE('Inserare reusita');*

*EXCEPTION*

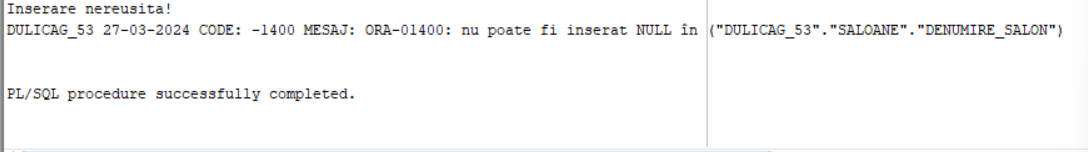
*WHEN exceptie THEN DBMS\_OUTPUT.PUT\_LINE('Inserare nereusita!');*

*DBMS\_OUTPUT.PUT\_LINE(USER||' '||SYSDATE||' CODE: '||SQLCODE||' MESAJ: '||SQLERRM);*

*END;*

*/*

**REZULTAT**



# 5. Proceduri, Funcții, Pachete

## 5.1. PROCEDURI

### 5.1.1.

**Să se mărească salariul standard al angajaților cu 10%, pentru cei care au avut un număr de ore suplimentare minim 4 și care au salariul mai mic decât media tutulor salariilor angajaților din toate saloanele. Se va face o procedură separată pentru calcularea mediei salariilor și se va afișa această medie.**

*CREATE OR REPLACE PROCEDURE salariul\_standard\_mediu(p\_salariul\_mediu OUT NUMBER)*

*IS*

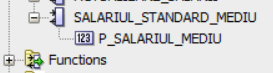
*BEGIN*

*SELECT AVG(salariu\_standard) INTO p\_salariul\_mediu FROM ANGAJATI\_SALON;*

*END;*

*/*



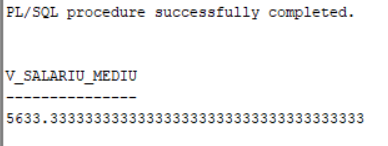


**APEL PROCEDURĂ 1:**

*VARIABLE g\_salariu\_mediu NUMBER*

*EXECUTE salariul\_standard\_mediu(:g\_salariu\_mediu)*

*PRINT g\_salariu\_mediu*

****

*CREATE OR REPLACE PROCEDURE modifica\_salariu (*

*p\_nr\_ore\_suplimentare IN angajati\_salon.numar\_ore\_suplimentare%TYPE,*

*marire IN NUMBER,*

*p\_salariu\_mediu IN NUMBER,*

*p\_nr\_mariri OUT NUMBER*

*)*

*IS*

*CURSOR ang\_cursor IS SELECT id\_angajat\_salon,nume\_angajat,salariu\_standard FROM ANGAJATI\_SALON WHERE id\_angajat\_salon IN (*

*SELECT ID\_ANGAJAT\_SALON*

*FROM ANGAJATI\_SALON*

*WHERE numar\_ore\_suplimentare >= p\_nr\_ore\_suplimentare*

*AND salariu\_standard < p\_salariu\_mediu*

*);*

*BEGIN*

*FOR ang\_rec IN ang\_cursor LOOP*

*UPDATE angajati\_salon*

*SET salariu\_standard = salariu\_standard \* (1 + marire / 100)*

*WHERE id\_angajat\_salon=ang\_rec.id\_angajat\_salon;*

*DBMS\_OUTPUT.PUT\_LINE('Modificare pentru angajatul ' || ang\_rec.nume\_angajat || ' cu salariul initial egal cu ' || ang\_rec.salariu\_standard);*

*END LOOP;*

*IF SQL%FOUND THEN p\_nr\_mariri:=SQL%ROWCOUNT;*

*ELSE*

*p\_nr\_mariri:=0;*

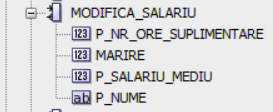
*END IF;*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN*

*DBMS\_OUTPUT.PUT\_LINE('Nu avem angajati cu numar de ore suplimentare mai mare sau egal cu ' || p\_nr\_ore\_suplimentare);*

*END;*

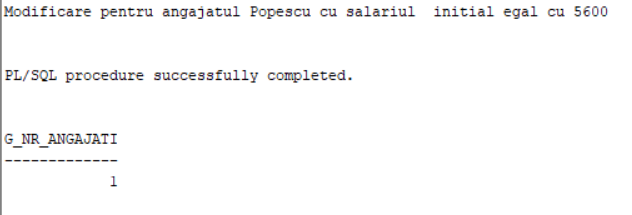
/

**APEL PROCEDURĂ 2:**

*VARIABLE g\_nr\_angajati NUMBER*

*EXECUTE modifica\_salariu(4,10,:g\_salariu\_mediu,:g\_nr\_angajati)*

*PRINT g\_nr\_angajati*



### 5.1.2.

**Să se afișeze clientul care a oferit cel mai mic rating pentru un anumit serviciu care a fost prestat în anul 2024. Se va implementa o procedură pentru a rezolva această problemă, iar procedura se va apela dintr-un bloc anonim. La final se va afisa un mesaj în care se va specifica (prenume client, serviciul prestat).**

*CREATE OR REPLACE PROCEDURE rating\_minim(p\_rating\_minim OUT NUMBER,p\_client OUT NUMBER,p\_an IN NUMBER, p\_stare\_programare IN VARCHAR)*

*IS*

*BEGIN*

*SELECT min(rating) INTO p\_rating\_minim FROM ISTORIC\_SERVICII;*

*SELECT p.id\_client\_salon INTO p\_client FROM programari p*

*JOIN istoric\_servicii i ON p.id\_client\_salon=i.id\_client\_salon*

*WHERE p.stare\_programare=p\_stare\_programare AND*

*extract(year from p.data\_programare)=p\_an AND i.rating=p\_rating\_minim;*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN*

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

*END;*

*/*



**APEL PROCEDURĂ :**

*DECLARE*

*v\_id\_client clienti\_salon.id\_client\_salon%TYPE;*

*v\_prenume\_client clienti\_salon.prenume\_client\_salon%TYPE;*

*v\_rating istoric\_servicii.rating%TYPE;*

*v\_serviciu servicii\_salon.denumire\_serviciu %TYPE;*

*BEGIN*

*rating\_minim(v\_rating,v\_id\_client,2024,'FINALIZATA');*

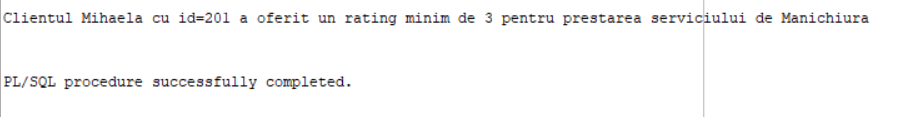
*SELECT prenume\_client\_salon INTO v\_prenume\_client FROM clienti\_salon WHERE id\_client\_salon=v\_id\_client;*

*SELECT s.denumire\_serviciu INTO v\_serviciu FROM istoric\_servicii i JOIN servicii\_salon s ON i.id\_serviciu=s.id\_serviciu WHERE i.rating=v\_rating;*

*DBMS\_OUTPUT.PUT\_LINE('Clientul '||v\_prenume\_client||' cu id='||v\_id\_client||' a oferit un rating minim de '||v\_rating||' pentru prestarea serviciului de '||v\_serviciu);*

*end;*

*/*



### 5.1.3.

**Să se creeze o procedură care să afișeze serviciul cu cel mai mare număr de programări și să se execute procedura pentru afișarea rezultatului.**

*CREATE OR REPLACE PROCEDURE cel\_mai\_utilizat\_serviciu( p\_serviciu\_maxim OUT SERVICII\_SALON.DENUMIRE\_SERVICIU%TYPE )*

*IS*

*BEGIN*

*SELECT s.DENUMIRE\_SERVICIU INTO p\_serviciu\_maxim*

*FROM PROGRAMARI p*

*JOIN SERVICII\_SALON s ON p.ID\_SERVICIU = s.ID\_SERVICIU*

*GROUP BY s.DENUMIRE\_SERVICIU*

*HAVING COUNT(p.ID\_CLIENT\_SALON) = (*

*SELECT MAX(COUNT(p.ID\_CLIENT\_SALON))*

*FROM PROGRAMARI p*

*JOIN SERVICII\_SALON s ON p.ID\_SERVICIU = s.ID\_SERVICIU*

*GROUP BY s.DENUMIRE\_SERVICIU);*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN*

*DBMS\_OUTPUT.PUT\_LINE('Nu exista programari pentru servicii.');*

*WHEN OTHERS THEN*

*DBMS\_OUTPUT.PUT\_LINE('A apărut o eroare');*

*END;*

/

**APEL PROCEDURĂ :**

*DECLARE*

*v\_serviciu\_maxim SERVICII\_SALON.DENUMIRE\_SERVICIU%TYPE;*

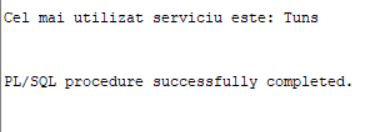
*BEGIN*

*cel\_mai\_utilizat\_serviciu(v\_serviciu\_maxim);*

*DBMS\_OUTPUT.PUT\_LINE('Cel mai utilizat serviciu este: ' || v\_serviciu\_maxim);*

*END;*

*/*



### 5.1.4.

**Utilizând tabela ANGAJATI\_SALON, să se implementeze o procedură care să îmi afișeze CALEA,NIVELUL pentru angajații direcți și indirecți ai angajatul cu id-ul 10. Pentru parcurgerea tuturor angajaților se va utiliza un cursor explicit. Apelați procedura într-un bloc PL/SQL.**

*CREATE OR REPLACE PROCEDURE AFISARE\_ANGAJATI*

*IS*

*CURSOR C\_ANGAJATI IS*

*SELECT LEVEL,*

*LPAD(' ', LEVEL) || NUME\_ANGAJAT AS nume,*

*SYS\_CONNECT\_BY\_PATH(NUME\_ANGAJAT, '->') AS cale*

*FROM ANGAJATI\_SALON*

*CONNECT BY PRIOR ID\_ANGAJAT\_SALON = ID\_MANAGER\_SALON*

*START WITH ID\_ANGAJAT\_SALON = 10*

*ORDER SIBLINGS BY NUME\_ANGAJAT;*

*V\_NIVEL NUMBER;*

*V\_POZ VARCHAR2(100);*

*V\_PATH VARCHAR2(200);*

*BEGIN*

*FOR REC IN C\_ANGAJATI LOOP*

*V\_NIVEL := REC.level;*

*V\_POZ := REC.nume;*

*V\_PATH := REC.cale;*

*DBMS\_OUTPUT.PUT\_LINE('Nivel: ' || V\_NIVEL || ', Nume : ' || V\_POZ || ' Cale: ' || V\_PATH);*

*END LOOP;*

*END;*

*/*

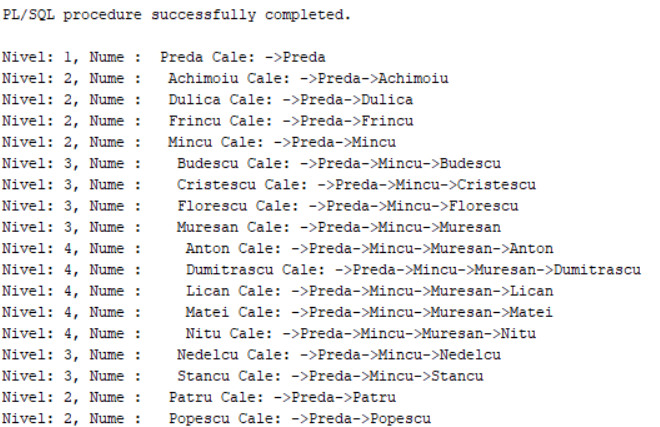
**APEL PROCEDURĂ :**

*BEGIN*

*AFISARE\_ANGAJATI;*

*END;*

*/*



## 5.2. FUNCȚII

### 5.2.1.

**Să se implementeze o funcție prin care inițial se aleg angajații care au un număr de ore suplimentare mai mare decât medie generala a orelor pentru toți angajații. După se va cauta in această listă dacă există vreun angajat care are un număr de ore suplimentare mai mare sau egal cu un număr introdus de la tastatură. Apelați această funcție și afișați un mesaj corespunzător.**

*CREATE OR REPLACE FUNCTION ore\_suplimentare(p\_nume\_angajat OUT ANGAJATI\_SALON.NUME\_ANGAJAT%TYPE,p\_ore OUT ANGAJATI\_SALON.numar\_ore\_suplimentare%TYPE, p\_medie IN NUMBER,p\_maxim IN NUMBER)*

*RETURN BOOLEAN*

*IS*

*CURSOR ang\_cursor IS SELECT nume\_angajat,numar\_ore\_suplimentare FROM ANGAJATI\_SALON*

*WHERE numar\_ore\_suplimentare >p\_medie;*

*v\_exista\_angajati BOOLEAN := FALSE;*

*BEGIN*

*FOR ang\_rec IN ang\_cursor LOOP*

*IF ang\_rec.numar\_ore\_suplimentare >= p\_maxim THEN*

*p\_nume\_angajat := ang\_rec.nume\_angajat;*

*p\_ore := ang\_rec.numar\_ore\_suplimentare;*

*v\_exista\_angajati := TRUE;*

*END IF;*

*END LOOP;*

*RETURN v\_exista\_angajati;*

*END;*

*/*



**APEL FUNCȚIE :**

*ACCEPT g\_max PROMPT 'Introduceti numarul maxim de ore'*

*DECLARE*

*v\_maxim NUMBER := &g\_max;*

*v\_medie NUMBER;*

*v\_nume ANGAJATI\_SALON.NUME\_ANGAJAT%TYPE;*

*v\_ore ANGAJATI\_SALON.numar\_ore\_suplimentare%TYPE;*

*BEGIN*

*SELECT AVG(NUMAR\_ORE\_SUPLIMENTARE)INTO v\_medie FROM ANGAJATI\_SALON;*

*IF(ore\_suplimentare(v\_nume,v\_ore,v\_medie,v\_maxim)) THEN DBMS\_OUTPUT.PUT\_LINE('Cel putin un angajat are ' || v\_maxim || ' ore suplimentare.');*

*DBMS\_OUTPUT.PUT\_LINE('Numele angajatului: ' || v\_nume || ', Ore suplimentare: ' || v\_ore);*

*ELSE*

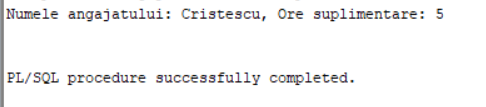
*DBMS\_OUTPUT.PUT\_LINE('Nu exista angajati cu un numar de ' || v\_maxim || ' ore suplimentare.');*

*END IF;*

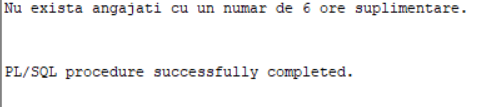
*END;*

*/*

*--PENTRU 5*



*--PENTRU 6*



### 5.2.2.

**Să se implementeze o funcție prin care să se afișeze angajatul care a avut cele mai multe programări între 1 august 2023 și 31 august 2024. Dacă în această perioada se constată că nicio programare nu s-a planificat, se va trata acest caz.**

*CREATE OR REPLACE FUNCTION eficienta\_angajat*

*RETURN angajati\_salon.nume\_angajat%TYPE*

*IS*

*CURSOR ang\_cursor IS*

*SELECT a.nume\_angajat FROM ANGAJATI\_SALON a*

*JOIN PROGRAMARI p ON a.ID\_ANGAJAT\_SALON = p.ID\_ANGAJAT\_SALON*

*WHERE p.data\_programare BETWEEN TO\_DATE('01-AUG-2023', 'DD-MON-YYYY') AND TO\_DATE('31-AUG-2024', 'DD-MON-YYYY')*

*GROUP BY a.nume\_angajat*

*ORDER BY COUNT(\*) DESC;*

*v\_nr NUMBER:=0;*

*v\_angajat angajati\_salon.nume\_angajat%TYPE;*

*BEGIN*

*FOR ang\_rec IN ang\_cursor LOOP*

*v\_angajat := ang\_rec.nume\_angajat;*

*v\_nr:=v\_nr+1;*

*exit when v\_nr=1;*

*end loop;*

*RETURN v\_angajat;*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN*

*DBMS\_OUTPUT.PUT\_LINE('Nu exista programari intre august 2023 si august 2024.');*

*RETURN NULL;*

*END;*

/



**APEL FUNCȚIE :**

*DECLARE*

*v\_nume angajati\_salon.nume\_angajat%TYPE;*

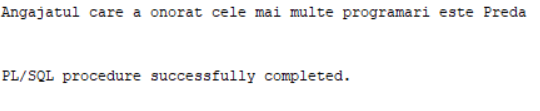
*BEGIN*

*v\_nume:=eficienta\_angajat;*

*DBMS\_OUTPUT.PUT\_LINE('Angajatul care a onorat cele mai multe programari este '||v\_nume);*

*END;*

*/*



### 5.2.3.

**Să se implementeze o funcție care să returneze media vârstei pentru clienții care vin la Manichiură și să se calculeze care este prețul mediu pe care aceștia îl plătesc pentru acest serviciu.**

*CREATE OR REPLACE FUNCTION media\_varsta\_si\_pret\_manichiura(p\_serviciu SERVICII\_SALON.DENUMIRE\_SERVICIU%TYPE, p\_media\_pret OUT NUMBER)*

*RETURN NUMBER*

*IS*

*v\_media\_varsta NUMBER;*

*v\_media\_pret NUMBER;*

*BEGIN*

*SELECT AVG(TRUNC(MONTHS\_BETWEEN(SYSDATE, data\_nasterii\_client) / 12)) INTO v\_media\_varsta*

*FROM CLIENTI\_SALON*

*WHERE ID\_CLIENT\_SALON IN (SELECT DISTINCT ID\_CLIENT\_SALON*

*FROM PROGRAMARI WHERE ID\_SERVICIU = (SELECT ID\_SERVICIU FROM SERVICII\_SALON WHERE DENUMIRE\_SERVICIU = p\_serviciu));*

*SELECT AVG(pret\_serviciu) INTO v\_media\_pret*

*FROM SERVICII\_SALON*

*WHERE DENUMIRE\_SERVICIU = p\_serviciu;*

*p\_media\_pret := v\_media\_pret;*

*RETURN v\_media\_varsta;*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN*

*DBMS\_OUTPUT.PUT\_LINE('Nu exista clienti pentru serviciul de manichiura.');*

*RETURN NULL;*

*END;*

*/*



**APEL FUNCȚIE :**

*DECLARE*

*v\_media\_varsta NUMBER;*

*v\_media\_pret NUMBER;*

*BEGIN*

*v\_media\_varsta := media\_varsta\_si\_pret\_manichiura('Manichiura', v\_media\_pret);*

*IF v\_media\_varsta IS NOT NULL THEN*

*DBMS\_OUTPUT.PUT\_LINE('Media varstei clientilor pentru manichiura este: ' || v\_media\_varsta);*

*DBMS\_OUTPUT.PUT\_LINE('Media pretului pentru serviciul de manichiura este: ' || v\_media\_pret);*

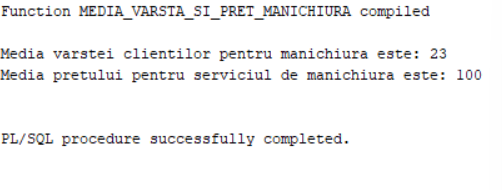
*ELSE*

*DBMS\_OUTPUT.PUT\_LINE('Nu exista clienti pentru serviciul de manichiura.');*

*END IF;*

*END;*

*/*



### 5.2.4.

**Se dorește implementarea unei funcții pentru a compara popularitatea a două servicii oferite în salonul de frumusețe. Funcția trebuie să determine care dintre cele două servicii, 'Serviciu1' și 'Serviciu2', este mai popular într-o anumită perioadă de timp (1 ianuarie 2023 și 31 august 2024). Servicii propuse: Tuns / Coafat.**

*CREATE OR REPLACE FUNCTION comparare\_popularitate\_servicii(p\_serviciu1 IN VARCHAR2, p\_serviciu2 IN VARCHAR2)*

*RETURN BOOLEAN*

*IS*

*v\_programari\_serviciu1 NUMBER;*

*v\_programari\_serviciu2 NUMBER;*

*BEGIN*

*SELECT COUNT(\*)*

*INTO v\_programari\_serviciu1*

*FROM PROGRAMARI p*

*JOIN SERVICII\_SALON s ON p.ID\_SERVICIU = s.ID\_SERVICIU*

*WHERE s.DENUMIRE\_SERVICIU = p\_serviciu1*

*AND p.data\_programare BETWEEN TO\_DATE('01-01-2023', 'DD-MM-YYYY') AND TO\_DATE('31-08-2024', 'DD-MM-YYYY');*

*SELECT COUNT(\*)*

*INTO v\_programari\_serviciu2*

*FROM PROGRAMARI p*

*JOIN SERVICII\_SALON s ON p.ID\_SERVICIU = s.ID\_SERVICIU*

*WHERE s.DENUMIRE\_SERVICIU = p\_serviciu2*

*AND p.data\_programare BETWEEN TO\_DATE('01-01-2023', 'DD-MM-YYYY') AND TO\_DATE('31-08-2024', 'DD-MM-YYYY');*

*IF v\_programari\_serviciu1 > v\_programari\_serviciu2 THEN*

*RETURN TRUE;*

*ELSIF v\_programari\_serviciu1 < v\_programari\_serviciu2 THEN*

*RETURN FALSE;*

*ELSE*

*RETURN NULL;*

*END IF;*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN*

*DBMS\_OUTPUT.PUT\_LINE('Nu exista programari pentru unul dintre servicii in perioada specificata');*

*RETURN NULL;*

*END;*

*/*



**APEL FUNCȚIE :**

*DECLARE*

*v\_popularitate BOOLEAN;*

*BEGIN*

*v\_popularitate := comparare\_popularitate\_servicii('Tuns', 'Coafat');*

*IF v\_popularitate IS NOT NULL THEN*

*IF v\_popularitate THEN*

*DBMS\_OUTPUT.PUT\_LINE('Serviciul Tuns este mai popular decat serviciul Coafat.');*

*ELSE*

*DBMS\_OUTPUT.PUT\_LINE('Serviciul Coafat este mai popular decat serviciul Tuns.');*

*END IF;*

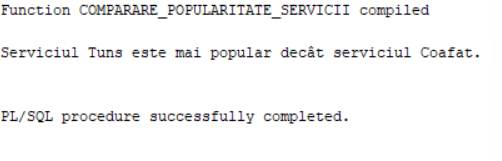
*ELSE*

*DBMS\_OUTPUT.PUT\_LINE('Nu exista suficiente date pentru a compara popularitatea serviciilor.');*

*END IF;*

*END;*

*/*



### 5.2.5.

**Să se afiseze timpul mediu pe care un client, care a avut mai mult de o programare, l-a petrecut în salon. Dacă nu se găsesc date, se va trata printr-o excepție.**

*CREATE OR REPLACE FUNCTION timp\_mediu\_petrecut(p\_numar\_programari IN NUMBER)*

*RETURN NUMBER*

*IS*

*v\_timp\_mediu NUMBER;*

*BEGIN*

*SELECT AVG(s.durata\_serviciu\_minute)*

*INTO v\_timp\_mediu*

*FROM programari p*

*JOIN servicii\_salon s ON p.id\_serviciu = s.id\_serviciu*

*WHERE p.id\_client\_salon IN (*

*SELECT id\_client\_salon FROM programari*

*GROUP BY id\_client\_salon*

*HAVING COUNT(\*) > p\_numar\_programari);*

*RETURN v\_timp\_mediu;*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN RETURN NULL;*

*END;*

*/*



**APEL FUNCȚIE :**

*DECLARE*

*v\_timp\_mediu NUMBER;*

*BEGIN*

*v\_timp\_mediu := timp\_mediu\_petrecut(1);*

*IF v\_timp\_mediu IS NOT NULL THEN*

*DBMS\_OUTPUT.PUT\_LINE('Timpul mediu petrecut in salon pentru clientii cu mai mult de o programare: ' || v\_timp\_mediu || ' minute.');*

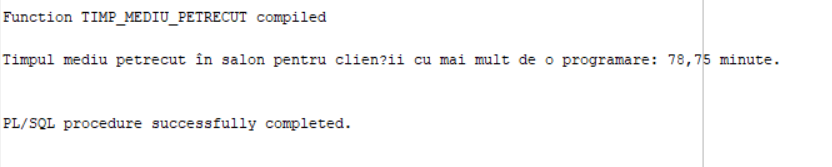
*ELSE*

*DBMS\_OUTPUT.PUT\_LINE('Nu exista suficiente date pentru a calcula timpul mediu petrecut in salon.');*

*END IF;*

*END;*

*/*



## 5.3. PACHETE

### 5.3.1

**Construiți un pachet care să conțină:**

**- o funcție care îmi afișează numele complet al unui angajat (nume+prenume), iar lânga acestea vor apărea și orașul în care lucrează angajatul respectiv, precum și numărul salonului din care face parte. (exemplu: Nume-Prenume-București2)**

**-o funcție care să returneze, pentru fiecare angajat, numărul de programări pe care acesta le-a efectuat. Există posibilitatea ca un angajat să fie nou, în cazul acesta nu a efectuat nicio programare. Se tratează printr-o excepție.**

**-o procedură care apelează funcțiile de mai sus și îmi va afișa doar angajații care au efectuat cele mai multe programări.**

**-se apelează procedura in cadrul pachetului**

*CREATE OR REPLACE PACKAGE pack\_angajati\_salon AS*

*FUNCTION date\_angajat(p\_id IN angajati\_salon.id\_angajat\_salon%TYPE) RETURN VARCHAR2;*

*FUNCTION numar\_programari(p\_id IN angajati\_salon.id\_angajat\_salon%TYPE) RETURN NUMBER;*

*PROCEDURE angajati\_prog\_min;*

*end;*

*/*

*CREATE OR REPLACE PACKAGE BODY pack\_angajati\_salon as*

*FUNCTION date\_angajat(p\_id IN angajati\_salon.id\_angajat\_salon%TYPE) RETURN VARCHAR2*

*IS*

*v\_nume angajati\_salon.nume\_angajat%TYPE;*

*v\_prenume angajati\_salon.prenume\_angajat%TYPE;*

*v\_oras saloane.oras%TYPE;*

*v\_nr saloane.id\_salon%TYPE;*

*v\_nume\_complet VARCHAR2(40);*

*BEGIN*

*SELECT a.nume\_angajat, a.prenume\_angajat,s.id\_salon,s.oras INTO v\_nume, v\_prenume,v\_nr,v\_oras FROM angajati\_salon a*

*JOIN saloane s ON a.id\_salon=s.id\_salon*

*WHERE a.id\_angajat\_salon=p\_id;*

*v\_nume\_complet:=v\_nume||' - '||v\_prenume||' - '||v\_oras||v\_nr;*

*RETURN v\_nume\_complet;*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN*

*RETURN NULL;*

*END;*

*FUNCTION numar\_programari(p\_id IN angajati\_salon.id\_angajat\_salon%TYPE) RETURN NUMBER*

*IS*

*v\_nr\_programari NUMBER;*

*angajat\_fara\_programari EXCEPTION;*

*BEGIN*

*SELECT COUNT(id\_angajat\_salon) INTO v\_nr\_programari FROM programari*

*WHERE id\_angajat\_salon=p\_id;*

*IF v\_nr\_programari=0 THEN*

*RAISE angajat\_fara\_programari;*

*ELSE*

*RETURN v\_nr\_programari;*

*END IF;*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN RETURN -2;*

*WHEN angajat\_fara\_programari THEN RETURN -1;*

*END;*

*PROCEDURE angajati\_prog\_min IS*

*CURSOR ang\_cursor IS SELECT id\_angajat\_salon, date\_angajat(id\_angajat\_salon), numar\_programari(id\_angajat\_salon) AS nr\_programari FROM angajati\_salon*

*ORDER BY nr\_programari DESC;*

*v\_max\_programari NUMBER;*

*BEGIN*

*SELECT MAX(numar\_programari(id\_angajat\_salon)) INTO v\_max\_programari FROM angajati\_salon;*

*FOR ang\_rec IN ang\_cursor LOOP*

*EXIT WHEN ang\_rec.nr\_programari<v\_max\_programari;*

*DBMS\_OUTPUT.PUT\_LINE('Angajatul '||date\_angajat(ang\_rec.id\_angajat\_salon)||' a efectuat '||ang\_rec.nr\_programari||' programari.');*

*end loop;*

*end;*

*END;*

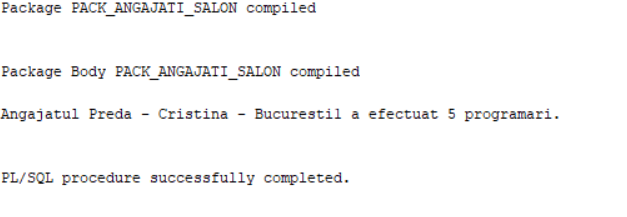
**APEL:**

*BEGIN*

*pack\_angajati\_salon.angajati\_prog\_min;*

*END;*

*/*



### 5.3.2

**Construiți un pachet care să conțină:**

**- o funcție care să îmi calculeze pentru fiecare client câte programări a FINALIZAT sau CONFIRMAT la salon. Se ține cont de faptul că unii clienți au ANULAT programarea, deci pot avea numarul de programări egal cu 0.**

**-o funcție care să returneze pentru toți clieții valoarea totală a serviciilor de care au beneficiat. Se va ține cont de faptul ca unii clienți au avut mai multe programări.**

**-o procedură care apelează funcțiile de mai sus și îmi va afișa primii 3 clienți care au plătit cel mai mult pentru a beneficia de aceste servicii.**

**-se apelează procedura in cadrul pachetului**

*CREATE OR REPLACE PACKAGE pack\_clienti\_fideli AS*

*FUNCTION nr\_programari\_clienti(p\_id\_client IN clienti\_salon.id\_client\_salon%TYPE) RETURN NUMBER;*

*FUNCTION valoare\_totala(p\_id\_client IN clienti\_salon.id\_client\_salon%TYPE) RETURN NUMBER;*

*PROCEDURE clienti\_fideli;*

*end;*

*/*

*CREATE OR REPLACE PACKAGE BODY pack\_clienti\_fideli as*

*FUNCTION nr\_programari\_clienti(p\_id\_client IN clienti\_salon.id\_client\_salon%TYPE) RETURN NUMBER*

*IS*

*v\_nr NUMBER;*

*programari\_anulate EXCEPTION;*

*BEGIN*

*SELECT COUNT(id\_client\_salon) INTO v\_nr FROM PROGRAMARI*

*WHERE id\_client\_salon=p\_id\_client AND stare\_programare IN ('FINALIZATA','CONFIRMATA');*

*IF v\_nr=0 THEN*

*RAISE programari\_anulate;*

*ELSE*

*RETURN v\_nr;*

*END IF;*

*EXCEPTION*

*WHEN programari\_anulate THEN RETURN 0;*

*END;*

*FUNCTION valoare\_totala(p\_id\_client IN clienti\_salon.id\_client\_salon%TYPE) RETURN NUMBER*

*IS*

*v\_total NUMBER;*

*BEGIN*

*SELECT SUM(S.PRET\_SERVICIU) SUMA\_TOTALA INTO v\_total*

*FROM SERVICII\_SALON S*

*JOIN ISTORIC\_SERVICII I*

*ON S.ID\_SERVICIU=I.ID\_SERVICIU*

*WHERE id\_client\_salon=p\_id\_client*

*GROUP BY I.ID\_CLIENT\_SALON;*

*RETURN v\_total;*

*END;*

*PROCEDURE clienti\_fideli*

*IS*

*CURSOR clienti\_cursor IS SELECT DISTINCT id\_client\_salon, nr\_programari\_clienti(id\_client\_salon), valoare\_totala(id\_client\_salon) AS valoare*

*FROM programari*

*ORDER BY valoare DESC*

*FETCH FIRST 3 rows only;*

*BEGIN*

*FOR clienti\_rec IN clienti\_cursor LOOP*

*DBMS\_OUTPUT.PUT\_LINE('Clientul cu id '||clienti\_rec.id\_client\_salon||' a beneficiat de '||nr\_programari\_clienti(clienti\_rec.id\_client\_salon)||' programari si a platit '||clienti\_rec.valoare||' lei in total');*

*END LOOP;*

*END;*

*end;*

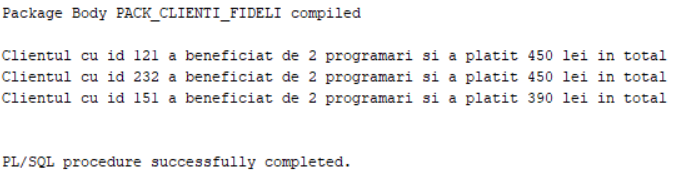
*/*

*BEGIN*

*pack\_clienti\_fideli.clienti\_fideli;*

*END;*

*/*



# 6. Triggeri

### 6.1.

**Implementați un trigger denumit ACTUALIZARI pentru a restricționa operațiile de inserare, actualizare și ștergere în tabela PROGRAMARI, astfel încât să nu se permită executarea pentru niciun fel de comandă, în afara programului de lucru. Utilizați funcțiile SYSDATE și CURRENT\_TIMESTAMP pentru a obține data și ora curentă. Apelați trigger-ul folosind o comandă la alegere.**

*CREATE OR REPLACE TRIGGER ACTUALIZARI*

*BEFORE INSERT OR DELETE OR UPDATE OF STARE\_PROGRAMARE ON PROGRAMARI*

*FOR EACH ROW*

*BEGIN*

*IF (TO\_CHAR(SYSDATE,'DY') NOT IN ('SAT','SUN')) OR (TO\_CHAR(SYSDATE,'HH24') NOT BETWEEN '08' AND '18') THEN*

*IF UPDATING THEN*

*RAISE\_APPLICATION\_ERROR(-20505,'ACTUALIZARILE SE POT FACE DOAR IN TIMPUL PROGRAMULUI DE LUCRU ! , DATA:'||SYSDATE||' '||CURRENT\_TIMESTAMP);*

*ELSIF INSERTING THEN*

*RAISE\_APPLICATION\_ERROR(-20500,'INSERARILE SE POT FACE DOAR IN TIMPUL PROGRAMULUI DE LUCRU ! , DATA:'||SYSDATE||' '||CURRENT\_TIMESTAMP);*

*ELSIF DELETING THEN*

*RAISE\_APPLICATION\_ERROR(-20510,'STERGERILE SE POT FACE DOAR IN TIMPUL PROGRAMULUI DE LUCRU ! , DATA:'||SYSDATE||' '||CURRENT\_TIMESTAMP);*

*END IF;*

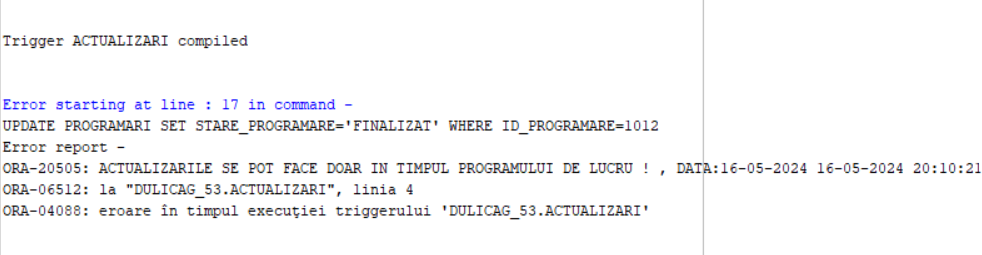
*END IF;*

*END;*

*/*

**VERIFICARE:**

*UPDATE PROGRAMARI SET STARE\_PROGRAMARE='FINALIZAT' WHERE ID\_PROGRAMARE=1012*



### 6.2.

**Implementați un trigger denumit AUDIT\_CLIENTI pentru a gestiona ștergerea programărilor anulate, respectiv a clienților care au anulat programarea. Să se creeze o tabelă separată în care se vor păstra date legate de id-ul clientului, cine a făcut ștergerea și data ștergerii.**

*CREATE TABLE audit\_clienti (*

*client\_id NUMBER(3),*

*utilizator VARCHAR2(100),*

*data\_stergere DATE*

*);*

*CREATE OR REPLACE TRIGGER AUDIT\_STERGERE\_CLIENTI*

*BEFORE DELETE ON PROGRAMARI*

*FOR EACH ROW*

*WHEN(OLD.STARE\_PROGRAMARE='ANULATA')*

*BEGIN*

*INSERT INTO audit\_clienti(client\_id, utilizator, data\_stergere)*

*VALUES (:OLD.id\_client\_salon, USER, SYSDATE);*

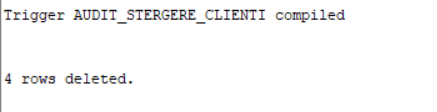
*END;*

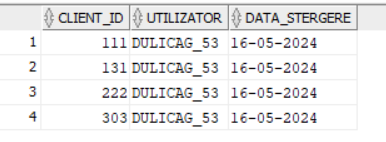
*/*

**VERIFICARE:**

*DELETE FROM PROGRAMARI WHERE STARE\_PROGRAMARE='ANULATA';*

*SELECT \* FROM AUDIT\_CLIENTI;*





### 6.3.

**Implementați un trigger denumit VEFICARE\_LIMITA\_SALARIU pentru a gestiona mărirea salariilor pentru angajații care nu au ore suplimentare, dar unde se ține cont de păstrarea unei limite a salariului de 6000 lei. Să se apeleze trigger-ul. (Exemplu: id\_angajat\_salon=23, angajat cu numar\_ore\_suplimentare=null și salariu inițial 4950 lei)**

*CREATE OR REPLACE TRIGGER VEFICARE\_LIMITA\_SALARIU*

*AFTER UPDATE OF SALARIU\_STANDARD ON ANGAJATI\_SALON*

*FOR EACH ROW*

*WHEN(NEW.NUMAR\_ORE\_SUPLIMENTARE IS NULL)*

*BEGIN*

*IF :NEW.SALARIU\_STANDARD>6000 THEN*

*RAISE\_APPLICATION\_ERROR(-20502,'FARA ORE SUPLIMENTARE, DEPASIRE LIMITA SALARIU');*

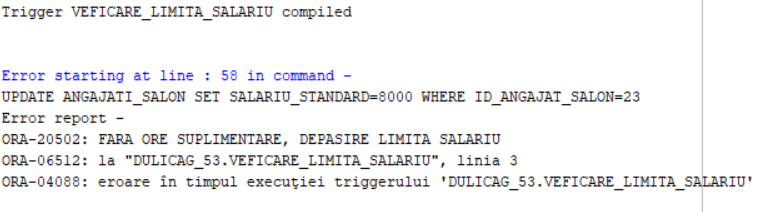
*END IF;*

*END;*

*/*

**VERIFICARE:**

UPDATE ANGAJATI\_SALON SET SALARIU\_STANDARD=8000 WHERE ID\_ANGAJAT\_SALON=23;



### 6.4.

**Implementați un trigger denumit VALIDARE\_NUME\_PRENUME pentru a gestiona clienții care se indentifica cu același nume. Dacă se încearcă să se introducă un client care are numele asemănător cu numele altui client deja existent în baza de date, lângă numele noului client se va pune o valoare (1,2..) în funcție de câte dubluri sunt găsite. Modificarea se va face în trigger.**

*CREATE OR REPLACE TRIGGER VALIDARE\_NUME\_PRENUME*

*BEFORE INSERT ON CLIENTI\_SALON*

*FOR EACH ROW*

*DECLARE*

*N NUMBER;*

*NEW\_NUME\_CLIENT\_SALON VARCHAR2(25);*

*COUNTER NUMBER := 1;*

*BEGIN*

*SELECT COUNT(\*) INTO N*

*FROM CLIENTI\_SALON*

*WHERE NUME\_CLIENT\_SALON = :NEW.NUME\_CLIENT\_SALON*

*AND PRENUME\_CLIENT\_SALON = :NEW.PRENUME\_CLIENT\_SALON;*

*IF N > 0 THEN*

*LOOP*

*NEW\_NUME\_CLIENT\_SALON := :NEW.NUME\_CLIENT\_SALON || ' ' || COUNTER;*

*SELECT COUNT(\*) INTO N*

*FROM CLIENTI\_SALON*

*WHERE NUME\_CLIENT\_SALON = NEW\_NUME\_CLIENT\_SALON*

*AND PRENUME\_CLIENT\_SALON = :NEW.PRENUME\_CLIENT\_SALON;*

*EXIT WHEN N = 0;*

*COUNTER := COUNTER + 1;*

*END LOOP;*

*:NEW.NUME\_CLIENT\_SALON := NEW\_NUME\_CLIENT\_SALON;*

*END IF;*

*END;*

*/*

**VERIFICARE:**

*INSERT INTO CLIENTI\_SALON (ID\_CLIENT\_SALON, NUME\_CLIENT\_SALON, PRENUME\_CLIENT\_SALON, TELEFON\_CLIENT\_SALON, SEX\_CLIENT, EMAIL\_CLIENT\_SALON, DATA\_NASTERII\_CLIENT)*

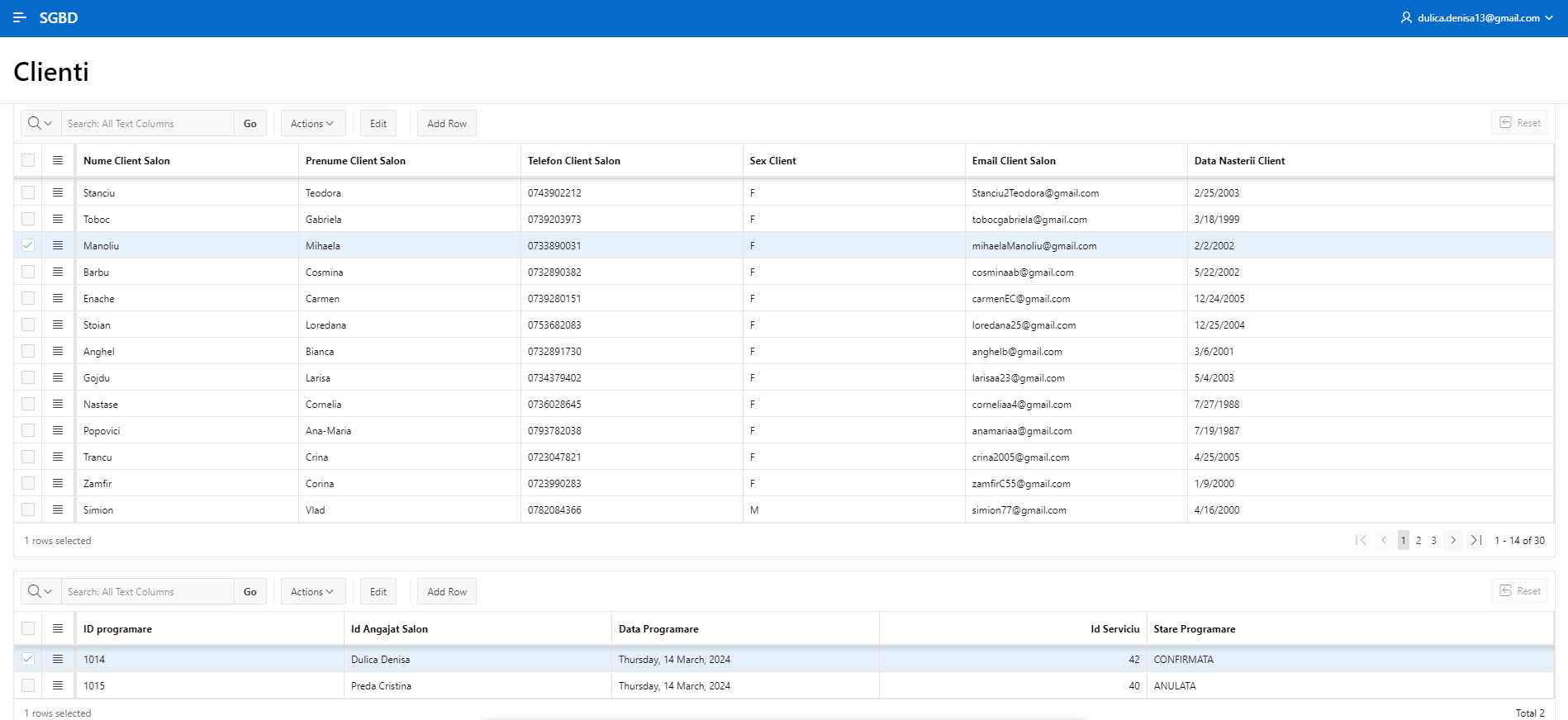
*VALUES (307, 'Anghel', 'Bianca', '0756389233', 'F', 'Biancaa91@gmail.com', TO\_DATE('06-09-2003', 'DD-MM-YYYY'));*

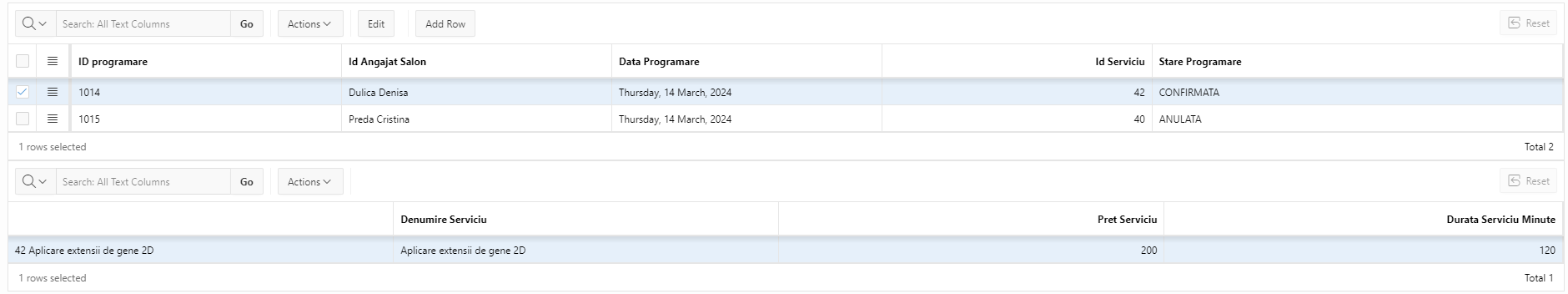


# 7. Interfața cu utilizatorul

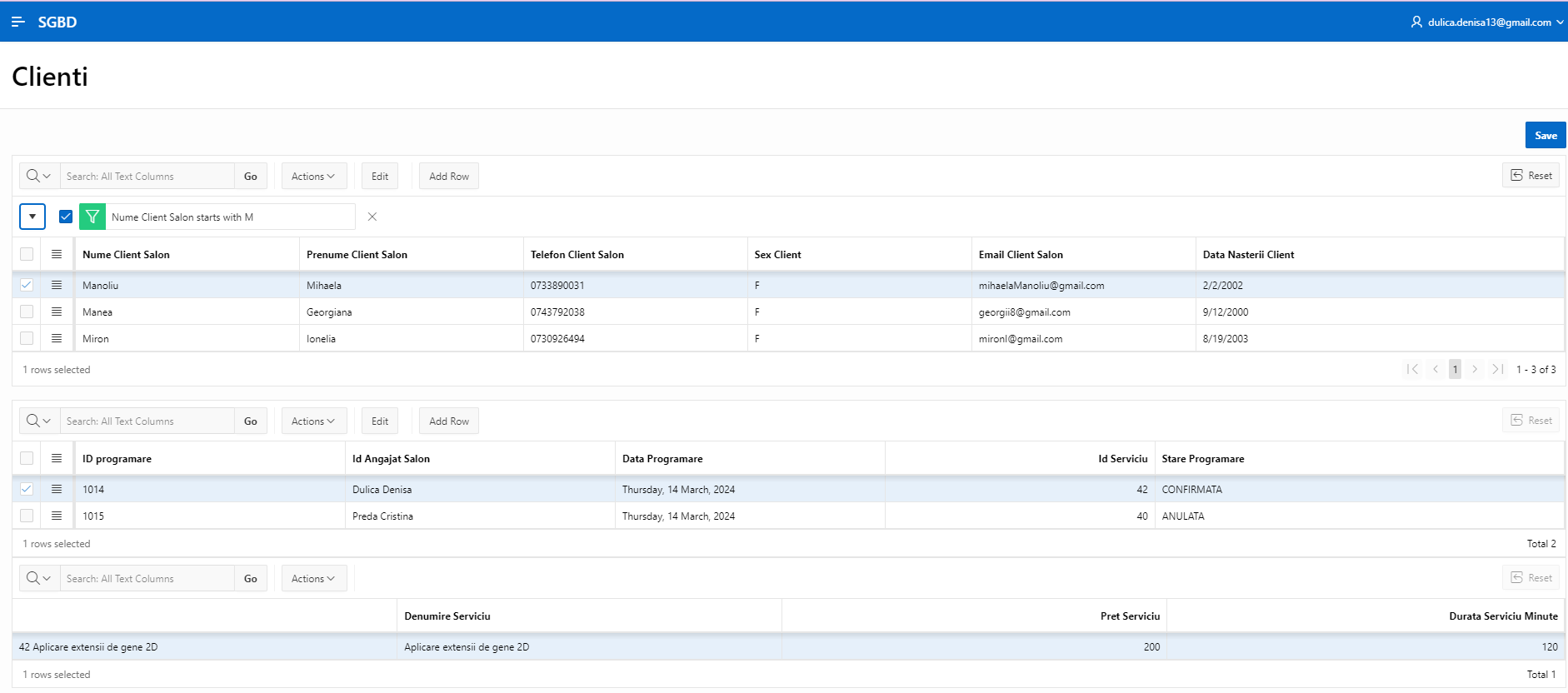
## 7.1. Master Detail

Utilizând baza de date a salonului, am creat o aplicație web prin modulul App Builder din Apex. Pentru interfața cu utilizatorul, am conceput inițial o pagină Master Detail de tip Stacked, numită Clienți, unde am implementat un raport detaliat despre clienții care au avut programări anulate, finalizate, sau confirmate. Fiecare client selectat în prima parte a tabelului va conduce la afișarea programărilor pe care acesta le-a efectuat, iar dacă selectăm, din a 2 a parte a tabelului, programarea efectuată, se vor afișa detalii legate de serviciul prestat precum denumire, prețul, durata.





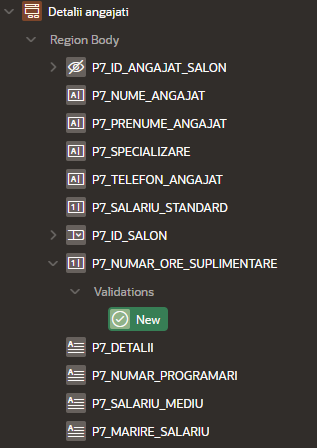
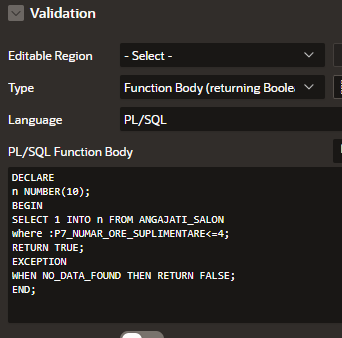
* Pentru a putea să vizualizăm mai clar datele, putem aplica filtre:

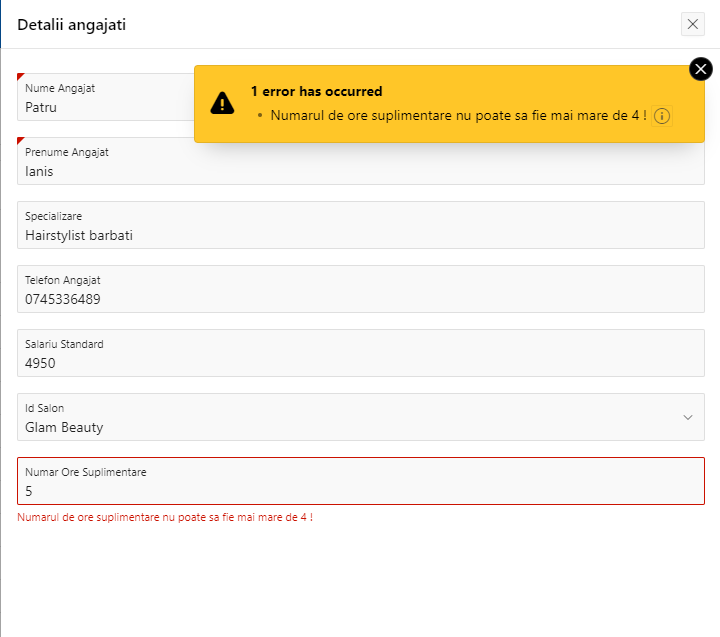


## 7.2. Set de pagini REPORT-FORM

Următorul raport din aplicație conține informații despre angajați, iar automat am creat și un formular de editare într-o pagină separată. În final am obținut 2 pagini, „Angajati”, fiind destinată raportului care conține datele din tabelă și următoarea „Detalii Angajati” în care se află formularul prin care pot să fac actualizări în baza de date .

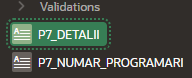
Pentru început am creat o validare în formular care împiedică introducerea unei valori mai mari de 4, pentru coloana NUMAR\_ORE\_SUPLIMENTARE.



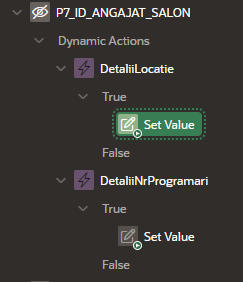
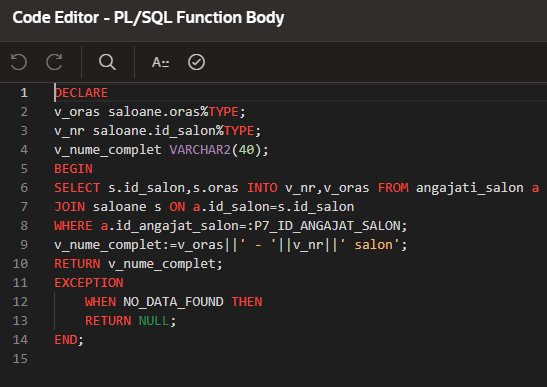
Creez o acțiune dinamică prin care, la selectarea unui angajat se vor completa automat căsuțele în care se specifică, în ce oraș lucrează angajatul și id-ul salonului, împreună cu numărul de programări pentru care angajatul a fost responsabil.

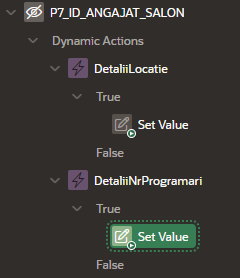
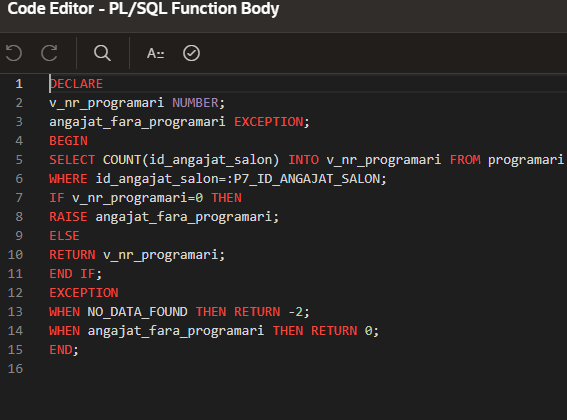
Am creat cele 2 item-uri noi în care se vor stoca informațiile.



Ulterior o să implementez acțiunile dinamice prin care se vor completa automat căsuțele noi ale angajatului selectat. Aceste acțiuni au proprietatea de a se declanșa în momentul în care, fiind în formular, vrem să vedem sau să modificăm informațiile despre un anumit angajat.

Pentru cele 2 item-uri, valorile pe care le vor primi sunt calculate prin aceste funcții:

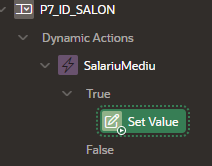
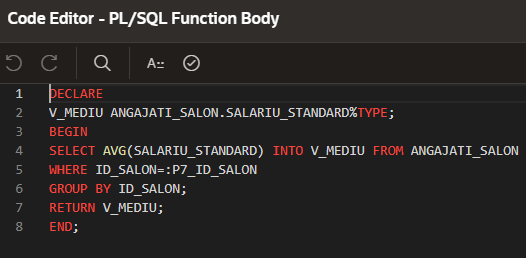
 

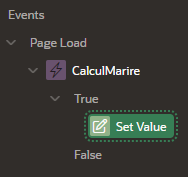
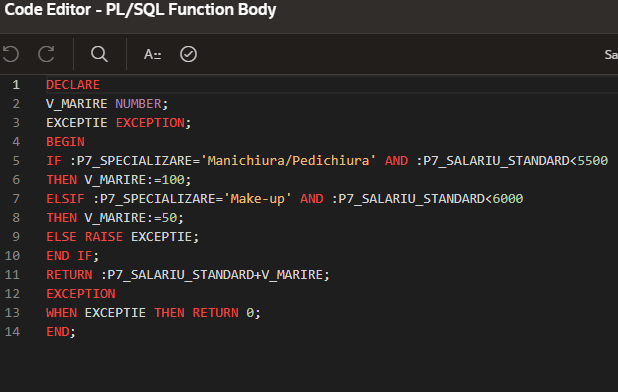
Am adaugat 2 item-uri noi:



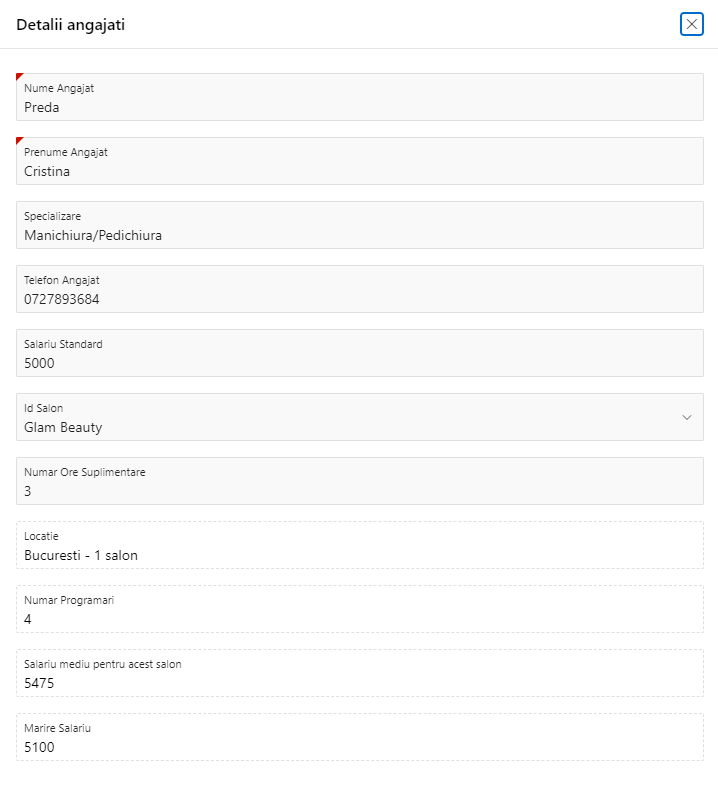
Un item în care am calculat salariul mediu al angajaților la nivelul salonului din care fac parte (exista 3 saloane în 3 orașe diferite).

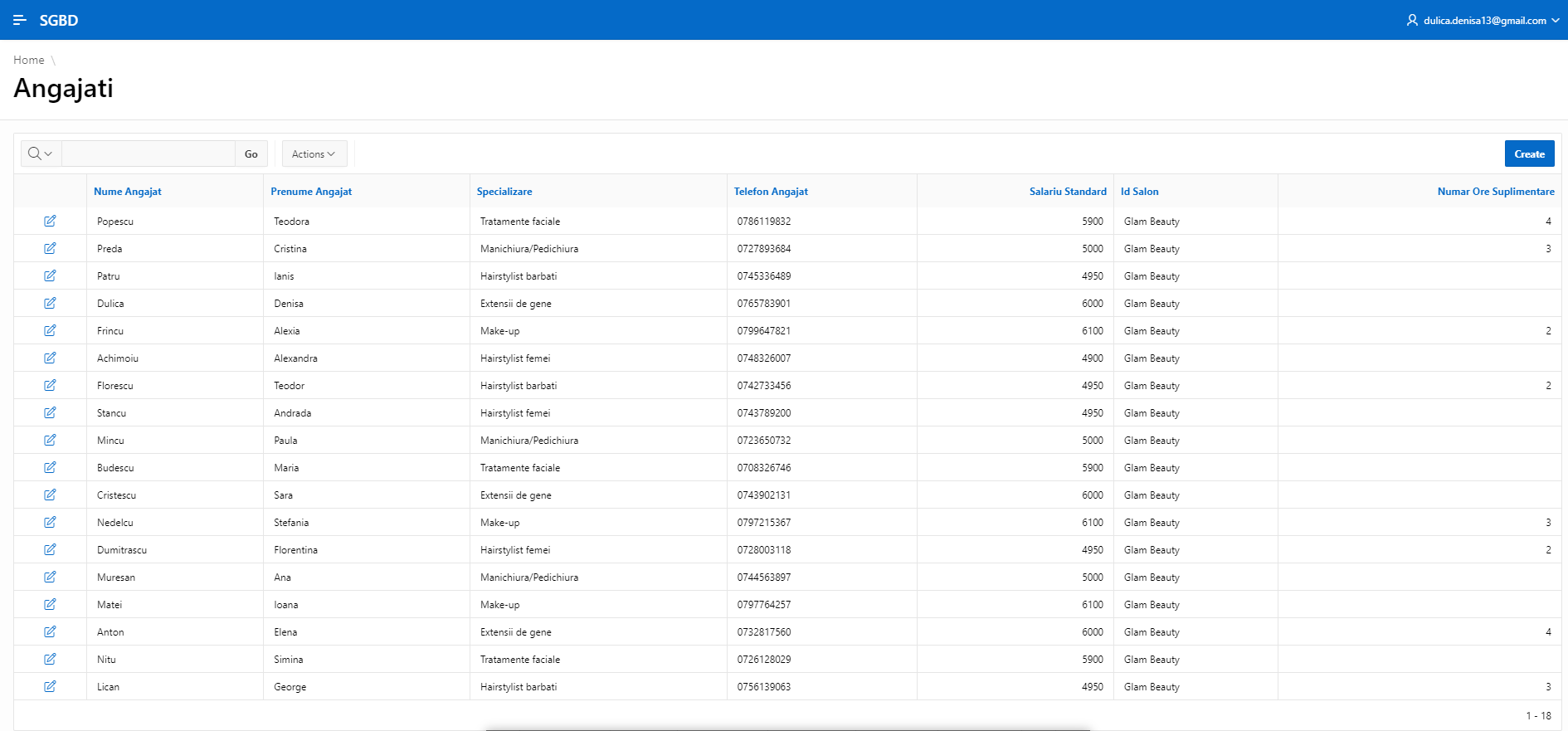
Și ultimul item introdus automat în formular, corespunde cu o acțiune dinamică prin care, la încărcarea paginii aferente formularului de actualizare a angajaților salonului, se va calcula automat o mărire a salariului pentru angajații care sunt specializați în Manichiură/Pedichiură (mărire cu 100) și Makeup (mărire cu 50) și care al căror salariu standard nu depășește o anumită limită.

Rezultat:



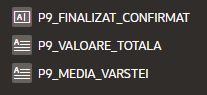
Formular:



## 7.3. Set 2 de pagini REPORT-FORM

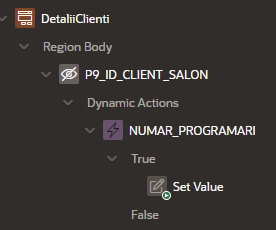
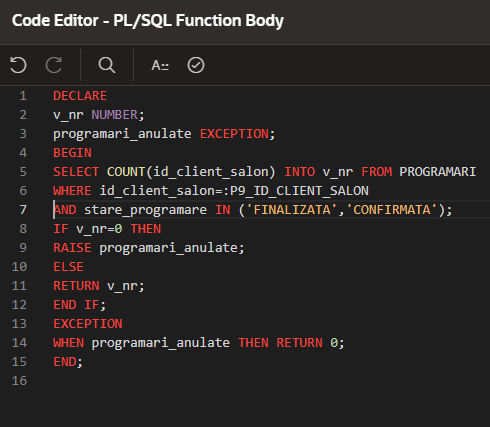
Acest raport conține informații despre clienții salonului, iar automat am creat și un formular de editare într-o pagină separată. În final am obținut 2 pagini, „Clienți”, fiind destinată raportului care conține datele din tabelă și următoarea „Detalii Clienți” în care se află formularul prin care pot să fac actualizări în baza de date .

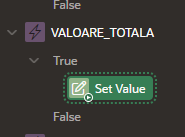
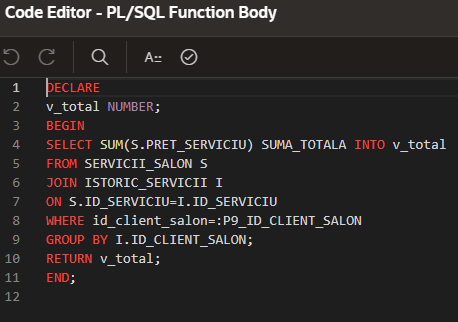
În primul rând, am stabilit cele 3 item-uri în care se vor stoca informațiile și care se vor introduce automat în formular.

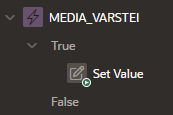
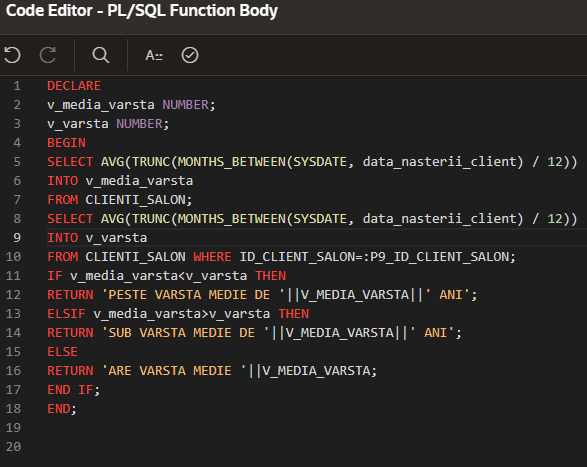


Am creat 3 acțiuni dinamice prin care, la selectarea unui client se vor completa automat item-urile legate de numărul de programări FINALIZATE/ CONFIRMATE pe care acesta le-a planificat la salon, valoarea totală a acestora și o informație legată de vârsta clientului raportată la vârsta medie a clienților care vin la salon. Aceste acțiuni au proprietatea de a se declanșa în momentul în care, fiind în formular, vrem să vedem sau să modificăm informațiile despre un anumit angajat pe care îl selectăm.

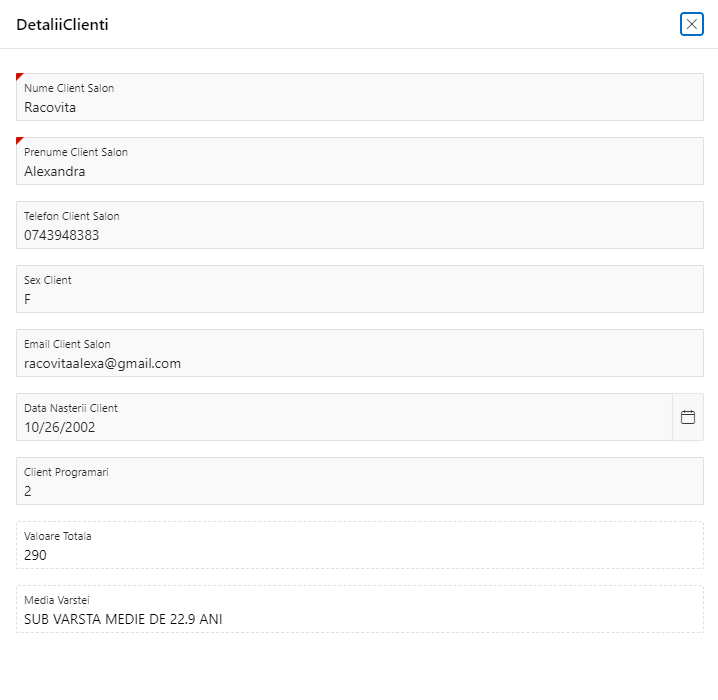
Pentru cele 3 item-uri, valorile pe care le vor primi sunt calculate prin aceste funcții:

Rezultat:



Formular:

