PROIECT SISTEME DE BAZE DE DATE

Aplicatia CareConnect

Student: Murariu Andrei-Stefanel

## Entitati si Tabele Asociative

* A) PACIENT
* B) PROGRAMARE
* C) MEDIC
* D) CONSULTATIE
* E) RETETA
* F) MEDICAMENT
* G) RETETA\_MEDICAMENT (Asociativa)
* H) ALERGIE
* I) PACIENT\_ALERGIE (Asociativa)
* J) DEPARTAMENT
* K) SPECIALIZARE
* L) MEDIC\_SPECIALIZARE (Asociativa)

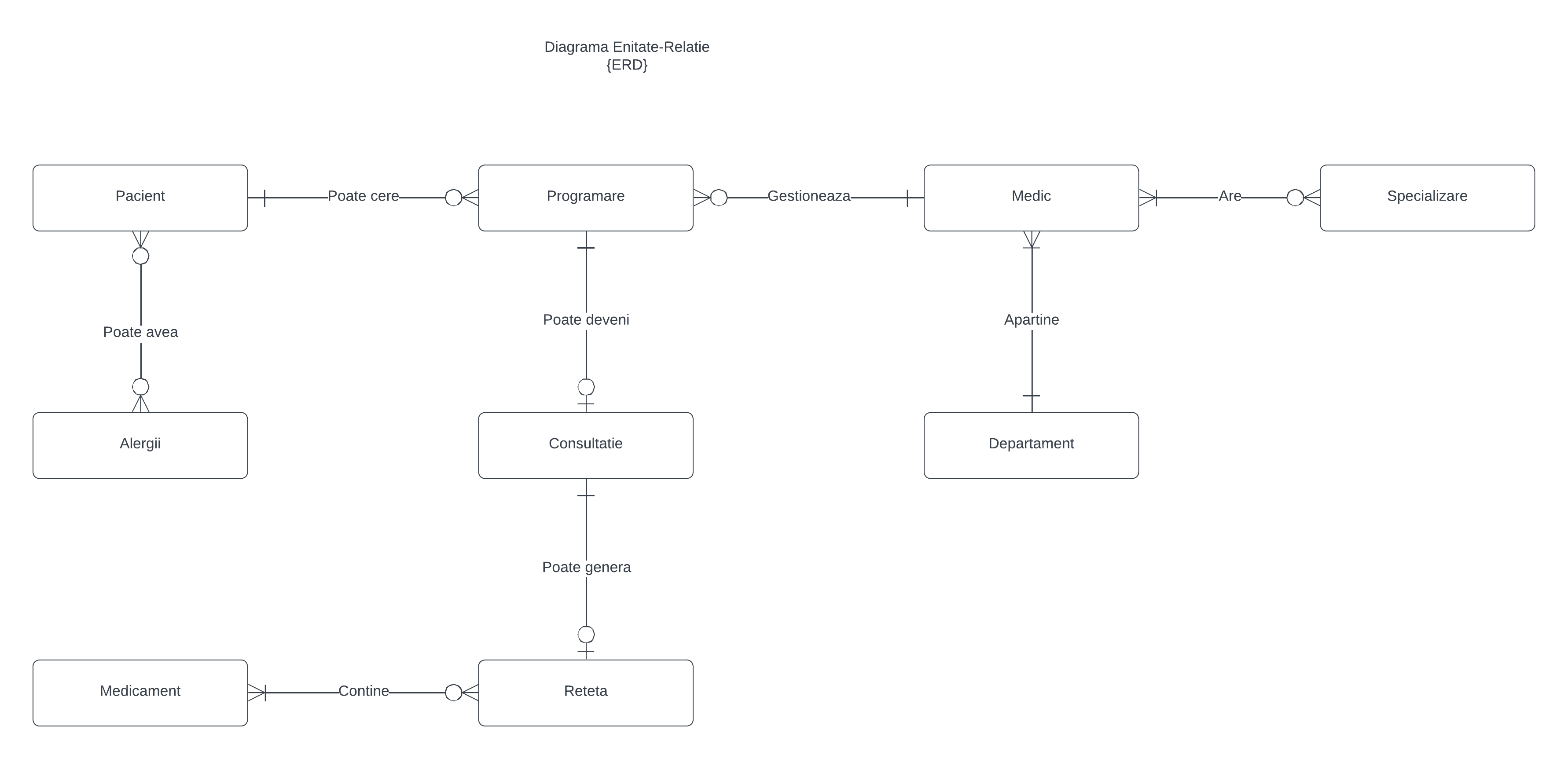
## Relatii

* PACIENT --(1:N)-- PROGRAMARE --(N:1)—MEDIC
* 2. PROGRAMARE --(1:1)—CONSULTATIE
* 3. MEDIC --(N:1)—DEPARTAMENT
* 4. CONSULTATIE --(1:1)—RETETA
* 5. RETETA --(1:N)-- RETETA\_MEDICAMENT --(N:1)—MEDICAMENT
* 6. MEDIC --(1:N)-- MEDIC\_SPECIALIZARE --(N:1)—SPECIALIZARE
* 7. PACIENT --(1:N)-- PACIENT\_ALERGIE --(N:1)—ALERGIE

## Constrângeri și Observații

* 1. ✅ PACIENT → PROGRAMARE → MEDIC
  + - Un pacient poate avea multiple programări
  + - Un medic poate avea multiple programări
  + - Fiecare programare trebuie să aibă exact un pacient și un medic
* 2. ✅ PROGRAMARE → CONSULTATIE
  + - O programare poate avea maxim o consultație
  + - O consultație trebuie să aparțină exact unei programări
* 3. ✅ MEDIC → DEPARTAMENT
  + - Un medic aparține exact unui departament
  + - Un departament poate avea mai mulți medici
* 4. ✅ CONSULTATIE → RETETA
  + - O consultație poate genera maxim o rețetă
  + - O rețetă aparține exact unei consultații
* 5. ✅ RETETA → RETETA\_MEDICAMENT → MEDICAMENT
  + - O rețetă poate conține multiple medicamente
  + - Un medicament poate apărea pe multiple rețete
  + - Fiecare asociere rețetă-medicament are propriile specificații (cantitate, dozaj)
* 6. ✅ MEDIC → MEDIC\_SPECIALIZARE → SPECIALIZARE
  + - Un medic poate avea multiple specializări
  + - O specializare poate fi deținută de mai mulți medici
  + - Fiecare asociere medic-specializare are data obținerii și număr certificat
* 7. ✅ PACIENT → PACIENT\_ALERGIE → ALERGIE
  + - Un pacient poate avea multiple alergii
  + - O alergie poate fi asociată mai multor pacienți
  + - Fiecare asociere pacient-alergie are propriul nivel de severitate
* 8. ✅ Constrângeri Generale
  + - Toate cheile externe trebuie să respecte integritatea referențială
  + - Datele temporale trebuie să fie valide și consistente
  + - Codurile unice (CNP, nr\_licenta, etc.) trebuie să fie unice în tabelele respective

## Diagrama ER (Entitate – Relatie)



## Diagrama Conceptuala

A screenshot of a computer flowchart

Description automatically generated

Diagrama conceptuală reprezintă o vedere de ansamblu a sistemului CareConnect, concentrându-se pe conceptele cheie și relațiile dintre ele.

## Design Logic

A screenshot of a computer screen

Description automatically generated

A diagram of a computer program

Description automatically generated with medium confidence

### Pacient

- Poate cere PROGRAMARE

- Poate avea ALERGII

- Istoricul medical este reflectat prin consultațiile și rețetele associate

Relatii Acxtualizate (mai in detaliu)

- 🏥 PACIENT ➡️ PROGRAMARE (0..N)

- Un pacient poate avea zero sau mai multe programări

- 🚨 PACIENT ➡️ ALERGII (0..N)

- Un pacient poate avea zero sau mai multe alergii înregistrate

Atribute

- ID - NUMBER(10) - PRIMARY KEY - NOT NULL - AUTOINCREMENT

- Nume - VARCHAR2(50) - NOT NULL

- Prenume - VARCHAR2(50) - NOT NULL

- CNP - VARCHAR2(13) - NOT NULL - CHECK (CNP LIKE '[1-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]') - UNIQUE

- Adresa - VARCHAR2(200) - NOT NULL

- Telefon - VARCHAR2(15) - NOT NULL - CHECK (Telefon LIKE '07%') - UNIQUE

- Email - VARCHAR2(100) - NOT NULL - CHECK (Email LIKE '%\_@\_\_%.\_\_%') - UNIQUE

- DataNasterii - DATE - NOT NULL - CHECK (DataNasterii < GETDATE())

- Sex - CHAR(1) - NOT NULL - CHECK (Sex IN ('M', 'F'))

- GrupaSanguina - VARCHAR2(3) - NOT NULL - CHECK (GrupaSanguina IN ('A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-'))

- DataInregistrarii - TIMESTAMP - NOT NULL - DEFAULT CURRENT\_TIMESTAMP

Constrangeri

- CNP este unic pentru fiecare pacient

- Telefonul este unic pentru fiecare pacient

- Emailul este unic pentru fiecare pacient

- DataNasterii trebuie sa fie mai mica decat data curenta

- Sexul trebuie sa fie M sau F

- GrupaSanguina trebuie sa fie A+ sau A- sau B+ sau B- sau AB+ sau AB- sau O+ sau O-

- DataInregistrarii este data curenta

### 6.2 Medic

- Gestionează PROGRAMĂRI

- Are SPECIALIZĂRI

- Aparține unui DEPARTAMENT

Relatii Acxtualizate (mai in detaliu)

- 📅 MEDIC ➡️ PROGRAMARE (0..N)

- Un medic poate gestiona zero sau mai multe programări

- 🎓 MEDIC ➡️ SPECIALIZARE (0..N) | SPECIALIZARE ➡️ MEDIC (1..N)

- Un medic poate avea zero sau mai multe specializări

- O specializare trebuie să fie deținută de cel puțin un medic

- 🏢 MEDIC ➡️ DEPARTAMENT (1..1) | DEPARTAMENT ➡️ MEDIC (1..N)

- Un medic aparține exact unui departament

- Un departament are cel puțin un medic

- 👑 DEPARTAMENT ➡️ MEDIC ȘEF (1..1)

- Un departament are exact un medic șef

Atribute

- ID - NUMBER(10) - PRIMARY KEY - NOT NULL - AUTOINCREMENT

- Nume - VARCHAR2(50) - NOT NULL

- Prenume - VARCHAR2(50) - NOT NULL

- CNP - VARCHAR2(13) - NOT NULL - CHECK (CNP LIKE '[1-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]') - UNIQUE

- Telefon - VARCHAR2(15) - NOT NULL - CHECK (Telefon LIKE '07%') - UNIQUE

- Email - VARCHAR2(100) - NOT NULL - CHECK (Email LIKE '%\_@\_\_%.\_\_%') - UNIQUE

- DataAngajare - DATE - NOT NULL - CHECK (DataAngajare <= GETDATE())

- GradProfesional - VARCHAR2(50) - NOT NULL

- NrLicenta - VARCHAR2(20) - NOT NULL - UNIQUE

- IdDepartament - NUMBER(10) - FOREIGN KEY - NOT NULL - REFERENCES Departament(ID)

Constrangeri

- DataAngajare trebuie sa fie mai mica sau egala cu data curenta

- CNP trebuie sa fie un numar unic pentru fiecare medic

- Telefonul trebuie sa fie un numar unic pentru fiecare medic

- Emailul trebuie sa fie un numar unic pentru fiecare medic

- NrLicenta trebuie sa fie un numar unic pentru fiecare medic

- IdDepartament trebuie sa fie un id valid pentru un departament

- GradProfesional trebuie sa fie o descriere a gradului profesional al medicului

### 6.3 Programare

- Este cerută de PACIENT

- Este gestionată de MEDIC

- Devine CONSULTAȚIE daca programarea nu este anulata

Atribute

- ID - NUMBER(10) - PRIMARY KEY - NOT NULL - AUTOINCREMENT

- IDPacient - NUMBER(10) - FOREIGN KEY - NOT NULL - REFERENCES Pacient(ID)

- IDMedic - NUMBER(10) - FOREIGN KEY - NOT NULL - REFERENCES Medic(ID)

- DataProgramare - DATE - NOT NULL - CHECK (DataProgramare >= GETDATE())

- OraProgramare - VARCHAR2(5) - NOT NULL - CHECK (OraProgramare LIKE '[0-9][0-9]:[0-9][0-9]')

- Status - VARCHAR2(20) - NOT NULL - CHECK (Status IN ('Programata', 'Anulata', 'Realizata'))

- MotivPrezentare - VARCHAR2(500) - NOT NULL

- Observatii - VARCHAR2(500)

Constrangeri

- DataProgramare trebuie sa fie mai mare sau egala cu data curenta

- OraProgramare trebuie sa fie in formatul HH:MM

- Statusul trebuie sa fie Programata, Anulata sau Realizata

- MotivPrezentare trebuie sa fie o descriere a motivului pentru care pacientul vine la programare

### 6.4 Departament

- Organizează MEDICI pe specialități

- Gestionează resurse și capacități

Atribute

- ID - NUMBER(10) - PRIMARY KEY - NOT NULL - AUTOINCREMENT

- NumeDepartament - VARCHAR2(100) - NOT NULL

- Locatie - VARCHAR2(100) - NOT NULL

- IdSefDepartament - NUMBER(10) - FOREIGN KEY - NOT NULL - REFERENCES Medic(ID)

- BugetAlocat - NUMBER(12, 2) - NOT NULL - CHECK (BugetAlocat > 0)

- NrPaturi - NUMBER(4) - NOT NULL - CHECK (NrPaturi > 0)

- TelefonContact - VARCHAR2(15) - NOT NULL - CHECK (TelefonContact LIKE '07%') - UNIQUE

Constrangeri

- NumeDepartament trebuie sa fie o descriere a departamentului

- Locatie trebuie sa fie o descriere a locatiei departamentului

- IdSefDepartament trebuie sa fie un id valid pentru un medic

- BugetAlocat trebuie sa fie un numar pozitiv

- NrPaturi trebuie sa fie un numar pozitiv

- TelefonContact trebuie sa fie un numar unic pentru fiecare departament

### 6.5 Consultatie

- Rezultă din PROGRAMARE

- Poate genera REȚETĂ

- 📋 PROGRAMARE ➡️ CONSULTAȚIE (0..1)

- O programare poate rezulta în zero sau o consultație

- 💊 CONSULTAȚIE ➡️ REȚETĂ (0..1)

- O consultație poate genera zero sau o rețetă

- 💊 REȚETĂ ➡️ MEDICAMENTE (1..N)

- O rețetă conține cel puțin un medicament

Atribute

- ID - NUMBER(10) - PRIMARY KEY - NOT NULL - AUTOINCREMENT

- IDProgramare - NUMBER(10) - FOREIGN KEY - NOT NULL - REFERENCES Programare(ID)

- DataConsultatie - TIMESTAMP - NOT NULL - DEFAULT CURRENT\_TIMESTAMP

- Diagnostic - VARCHAR2(500) - NOT NULL

- Observatii - VARCHAR2(500)

- Recomandari - VARCHAR2(500)

- Urgenta - NUMBER(1) - NOT NULL - CHECK (Urgenta IN (1, 2, 3, 4, 5))

Constrangeri

- DataConsultatie trebuie sa fie data curenta

- Urgenta trebuie sa fie un numar intre 1 si 5

### 6.6 Alergie

- Sunt asociate PACIENTULUI

- Importante pentru siguranța tratamentului

Atribute

- IDPacient - NUMBER(10) - FOREIGN KEY - NOT NULL - REFERENCES Pacient(ID)

- IDAlergie - NUMBER(10) - FOREIGN KEY - NOT NULL - REFERENCES Alergie(ID)

- DataDiagnostic - DATE - NOT NULL - CHECK (DataDiagnostic <= GETDATE())

- Observatii - VARCHAR2(500)

- Severitate - NUMBER(1) - NOT NULL - CHECK (Severitate IN (1, 2, 3, 4, 5))

Constrangeri

- DataDiagnostic trebuie sa fie mai mica sau egala cu data curenta

- Severitatea trebuie sa fie un numar intre 1 si 5

### 6.7 Medicament

- Este inclus în REȚETĂ

- Are specificații și instrucțiuni

Atribute

- ID - NUMBER(10) - PRIMARY KEY - NOT NULL - AUTOINCREMENT

- Denumire - VARCHAR2(100) - NOT NULL

- SubstantaActiva - VARCHAR2(200) - NOT NULL

- Concentratie - VARCHAR2(50) - NOT NULL - CHECK (Concentratie LIKE '[0-9][0-9]%')

- FormaFarmaceutica - VARCHAR2(50) - NOT NULL

- Producator - VARCHAR2(100) - NOT NULL

- PretUnitar - NUMBER(10, 2) - NOT NULL - CHECK (PretUnitar > 0)

- StocDisponibil - NUMBER(10) - NOT NULL - CHECK (StocDisponibil >= 0)

- NecesitaReteta - NUMBER(1) - NOT NULL - CHECK (NecesitaReteta IN (0, 1))

Constrangeri

- Pretul unitar trebuie sa fie un numar pozitiv

- Stocul disponibil trebuie sa fie un numar pozitiv

- NecesitaReteta trebuie sa fie 0 sau 1

### 6.8 Reteta

- Este generată din CONSULTAȚIE

- Conține MEDICAMENTE

Atribute

- ID - NUMBER(10) - PRIMARY KEY - NOT NULL - AUTOINCREMENT

- IDConsultatie - NUMBER(10) - FOREIGN KEY - NOT NULL - REFERENCES Consultatie(ID)

- DataPrescriere - TIMESTAMP - NOT NULL - DEFAULT CURRENT\_TIMESTAMP

- DataExpirare - TIMESTAMP - NOT NULL - CHECK (DataExpirare > DataPrescriere)

- Observatii - VARCHAR2(500)

- Status - VARCHAR2(20) - NOT NULL - CHECK (Status IN ('In asteptare', 'In curs', 'Finalizata'))

- CodUnic - VARCHAR2(20) - NOT NULL - UNIQUE

Constrangeri

- DataExpirare trebuie sa fie mai mare decat data prescrierii

- Statusul trebuie sa fie In asteptare, In curs sau Finalizata

- Codul unic trebuie sa fie un numar unic pentru fiecare reteta

### 6.9 Specializare

- Este deținută de MEDIC

- Definește aria de expertiză

Atribute

- IDMedic - NUMBER(10) - FOREIGN KEY - NOT NULL - REFERENCES Medic(ID)

- IDSpecializare - NUMBER(10) - FOREIGN KEY - NOT NULL - REFERENCES Specializare(ID)

- DataObtinere - DATE - NOT NULL - CHECK (DataObtinere <= GETDATE())

- CertificatNr - VARCHAR2(20) - NOT NULL - UNIQUE

Constrangeri

- DataObtinerii trebuie sa fie mai mica sau egala cu data curenta

- Certificatul trebuie sa fie un numar unic pentru fiecare medic si specializare

## Design Fizic

A screenshot of a computer

Description automatically generated

**Exemplu: Specializările unui Medic**

În modelul inițial, un medic ar putea avea mai multe specializări stocate într-un singur câmp, ceea ce încalcă principiul atomicității (FN1):

**Tabel MEDIC (neformal, înainte de normalizare):**

| **id\_medic** | **nume** | **specializari** |
| --- | --- | --- |
| 1 | Dr. Pop | cardiologie, pediatrie |
| 2 | Dr. Ion | neurologie, psihiatrie, geriatrie |

**Probleme identificate:**

* Câmpul 'specializari' conține multiple valori
* Nu se pot face căutări eficiente după o specializare specifică
* Dificultate în adăugarea sau eliminarea unei specializări
* Imposibilitatea stocării informațiilor adiționale despre specializare

**Soluția (FN1):**

**1. Crearea tabelei SPECIALIZARE:**

| **id\_specializare** | **denumire** | **descriere** | **nivel** |
| --- | --- | --- | --- |
| 1 | cardiologie | ... | senior |
| 2 | pediatrie | ... | primar |

**2. Crearea tabelei de joncțiune MEDIC\_SPECIALIZARE:**

| **id\_medic** | **id\_specializare** | **data\_obtinere** | **nr\_certificat** |
| --- | --- | --- | --- |
| 1 | 1 | 2020-01-01 | CARD2020/001 |
| 1 | 2 | 2018-06-15 | PED2018/123 |

Această soluție:

* Elimină atributul multivaloare (specializări) din tabela MEDIC
* Creează o tabelă separată pentru SPECIALIZARE cu propriile atribute
* Folosește o tabelă de joncțiune pentru a gestiona relația many-to-many
* Permite stocarea informațiilor adiționale despre fiecare specializare a medicului (data\_obtinere, nr\_certificat)

**Tabel în FN1 dar nu în FN2**

**Exemplu: Consultație**

CONSULTATIE(id\_consultatie, id\_programare, nume\_medic, specializare\_medic, grad\_medic, diagnostic, observatii)

**Probleme identificate:**

1. Dependențe parțiale:
   * nume\_medic → id\_medic (vine din programare)
   * specializare\_medic → id\_medic
   * grad\_medic → id\_medic
2. Informațiile despre medic sunt redundante și pot deveni inconsistente

**Soluția (FN2):**

1. Tabel CONSULTATIE: CONSULTATIE(id\_consultatie, id\_programare, diagnostic, observatii)
2. Informațiile despre medic sunt păstrate în tabelul MEDIC și accesate prin id\_medic din PROGRAMARE: MEDIC(id\_medic, nume\_medic, specializare\_medic, grad\_medic)

**Tabel în FN2 dar nu în FN3**

**Exemplu: Departament**

**Tabel inițial (în FN2 dar nu în FN3):**

DEPARTAMENT(id\_departament, nume\_departament, id\_sef\_departament, nume\_sef, grad\_sef, telefon\_sef, email\_sef, data\_numire\_sef)

**Probleme identificate - Dependențe tranzitive:**

1. id\_departament → id\_sef\_departament → nume\_sef
2. id\_departament → id\_sef\_departament → grad\_sef
3. id\_departament → id\_sef\_departament → telefon\_sef
4. id\_departament → id\_sef\_departament → email\_sef

Această structură creează:

* Redundanță în date
* Risc de inconsistență
* Dificultăți la actualizare

**Soluția (FN3):**

1. Tabel DEPARTAMENT (informații de bază): DEPARTAMENT(id\_departament, nume\_departament, id\_sef\_departament)
2. Informațiile despre șef sunt păstrate în tabelul MEDIC: MEDIC(id\_medic, nume, grad, telefon, email)

## Creare Baza de Date si Populare

**🏥 CareConnect - Implementare Bază de Date Oracle**

**🔄 Ordinea de Implementare**

**1️. Creare User și Tablespace (1\_creare.sql)**

* Creare user ddsys cu drepturi necesare
* Creare tablespace-uri separate pentru date și indexuri:
  + careconnect\_data: 100MB pentru date
  + careconnect\_index: 50MB pentru indexuri
* ⚠️ Important: Drepturile necesare trebuie acordate înainte de crearea obiectelor

-- Oracle docker - sysdba

CREATE USER ddsys IDENTIFIED BY 'password';

GRANT CONNECT, RESOURCE TO ddsys;

SELECT username FROM all\_users WHERE username = 'DDSYS';

GRANT CREATE USER TO ddsys;

GRANT CREATE TABLESPACE TO ddsys;

GRANT UNLIMITED TABLESPACE TO ddsys;

GRANT CREATE SESSION TO ddsys;

GRANT CREATE TABLE TO ddsys;

GRANT CREATE SEQUENCE TO ddsys;

-- =======================================

-- DataGrip

SELECT USER FROM DUAL;

CREATE TABLESPACE careconnect\_data

DATAFILE 'careconnect\_data.dbf'

SIZE 100M

AUTOEXTEND ON NEXT 50M;

CREATE TABLESPACE careconnect\_index

DATAFILE 'careconnect\_index.dbf'

SIZE 50M

AUTOEXTEND ON NEXT 25M;

**2️. Creare Tabele (2\_tables.sql)**

Ordinea de creare este importantă datorită dependențelor:

1. Tabele independente (fără FK):
   * PACIENT
   * DEPARTAMENT
   * SPECIALIZARE
   * ALERGIE
   * MEDICAMENT
2. Tabele cu dependențe simple:
   * MEDIC (depinde de DEPARTAMENT)
   * PROGRAMARE (depinde de PACIENT și MEDIC)
   * CONSULTATIE (depinde de PROGRAMARE)
   * RETETA (depinde de CONSULTATIE)
3. Tabele asociative (many-to-many):
   * RETETA\_MEDICAMENT
   * MEDIC\_SPECIALIZARE
   * PACIENT\_ALERGIE
4. Constrângeri circulare:
   * Relația MEDIC-DEPARTAMENT (șef departament) necesită ALTER TABLE

SAVEPOINT start\_transaction;

CREATE TABLE PACIENT (

ID NUMBER(10) PRIMARY KEY,

Nume VARCHAR2(50) NOT NULL,

Prenume VARCHAR2(50) NOT NULL,

CNP VARCHAR2(13) NOT NULL,

Adresa VARCHAR2(200) NOT NULL,

Telefon VARCHAR2(15) NOT NULL,

Email VARCHAR2(100) NOT NULL,

DataNasterii DATE NOT NULL,

Sex CHAR(1) NOT NULL,

GrupaSanguina VARCHAR2(3) NOT NULL,

DataInregistrarii TIMESTAMP DEFAULT CURRENT\_TIMESTAMP NOT NULL,

-- Unique constraints

CONSTRAINT UQ\_PACIENT\_CNP UNIQUE (CNP),

CONSTRAINT UQ\_PACIENT\_TELEFON UNIQUE (Telefon),

CONSTRAINT UQ\_PACIENT\_EMAIL UNIQUE (Email),

-- Check constraints

CONSTRAINT CHK\_PACIENT\_CNP CHECK (REGEXP\_LIKE(CNP, '^[1-9][0-9]{12}$')),

CONSTRAINT CHK\_PACIENT\_TELEFON CHECK (REGEXP\_LIKE(Telefon, '^07[0-9]{8}$')),

CONSTRAINT CHK\_PACIENT\_EMAIL CHECK (REGEXP\_LIKE(Email, '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$')),

CONSTRAINT CHK\_PACIENT\_SEX CHECK (Sex IN ('M', 'F')),

CONSTRAINT CHK\_PACIENT\_GRUPA\_SANGUINA CHECK (GrupaSanguina IN ('A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-'))

);

CREATE TABLE SPECIALIZARE (

ID NUMBER(10) PRIMARY KEY,

Denumire VARCHAR2(100) NOT NULL,

Descriere VARCHAR2(500) NOT NULL,

Nivel VARCHAR2(50) NOT NULL,

CodSpecializare VARCHAR2(20) NOT NULL,

-- Unique constraint

CONSTRAINT UQ\_SPECIALIZARE\_COD UNIQUE (CodSpecializare)

);

CREATE TABLE DEPARTAMENT (

ID NUMBER(10) PRIMARY KEY,

NumeDepartament VARCHAR2(100) NOT NULL,

Locatie VARCHAR2(100) NOT NULL,

IdSefDepartament NUMBER(10),

BugetAlocat NUMBER(12,2) NOT NULL,

NrPaturi NUMBER(4) NOT NULL,

TelefonContact VARCHAR2(15) NOT NULL,

-- Unique constraint

CONSTRAINT UQ\_DEPARTAMENT\_TELEFON UNIQUE (TelefonContact),

-- Check constraints

CONSTRAINT CHK\_DEPARTAMENT\_BUGET CHECK (BugetAlocat > 0),

CONSTRAINT CHK\_DEPARTAMENT\_PATURI CHECK (NrPaturi > 0),

CONSTRAINT CHK\_DEPARTAMENT\_TELEFON CHECK (REGEXP\_LIKE(TelefonContact, '^07[0-9]{8}$'))

);

CREATE TABLE MEDIC (

ID NUMBER(10) PRIMARY KEY,

Nume VARCHAR2(50) NOT NULL,

Prenume VARCHAR2(50) NOT NULL,

CNP VARCHAR2(13) NOT NULL,

Telefon VARCHAR2(15) NOT NULL,

Email VARCHAR2(100) NOT NULL,

DataAngajare DATE NOT NULL,

GradProfesional VARCHAR2(50) NOT NULL,

NrLicenta VARCHAR2(20) NOT NULL,

IdDepartament NUMBER(10) NOT NULL,

-- Unique constraints

CONSTRAINT UQ\_MEDIC\_CNP UNIQUE (CNP),

CONSTRAINT UQ\_MEDIC\_TELEFON UNIQUE (Telefon),

CONSTRAINT UQ\_MEDIC\_EMAIL UNIQUE (Email),

CONSTRAINT UQ\_MEDIC\_NR\_LICENTA UNIQUE (NrLicenta),

-- Foreign key constraint

CONSTRAINT FK\_MEDIC\_DEPARTAMENT FOREIGN KEY (IdDepartament)

REFERENCES DEPARTAMENT(ID),

-- Check constraints

CONSTRAINT CHK\_MEDIC\_CNP CHECK (REGEXP\_LIKE(CNP, '^[1-9][0-9]{12}$')),

CONSTRAINT CHK\_MEDIC\_TELEFON CHECK (REGEXP\_LIKE(Telefon, '^07[0-9]{8}$')),

CONSTRAINT CHK\_MEDIC\_EMAIL CHECK (REGEXP\_LIKE(Email, '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$'))

);

ALTER TABLE DEPARTAMENT

ADD CONSTRAINT FK\_DEPARTAMENT\_SEF FOREIGN KEY (IdSefDepartament)

REFERENCES MEDIC(ID);

CREATE TABLE PROGRAMARE (

ID NUMBER(10) PRIMARY KEY,

IDPacient NUMBER(10) NOT NULL,

IDMedic NUMBER(10) NOT NULL,

DataProgramare DATE NOT NULL,

OraProgramare VARCHAR2(5) NOT NULL,

Status VARCHAR2(20) NOT NULL,

MotivPrezentare VARCHAR2(500) NOT NULL,

Observatii VARCHAR2(500),

-- Foreign key constraints

CONSTRAINT FK\_PROGRAMARE\_PACIENT FOREIGN KEY (IDPacient)

REFERENCES PACIENT(ID),

CONSTRAINT FK\_PROGRAMARE\_MEDIC FOREIGN KEY (IDMedic)

REFERENCES MEDIC(ID),

-- Check constraints

CONSTRAINT CHK\_PROGRAMARE\_ORA CHECK (REGEXP\_LIKE(OraProgramare, '^([0-1][0-9]|2[0-3]):[0-5][0-9]$')),

CONSTRAINT CHK\_PROGRAMARE\_STATUS CHECK (Status IN ('Programata', 'Anulata', 'Realizata'))

);

CREATE TABLE ALERGIE (

ID NUMBER(10) PRIMARY KEY,

Denumire VARCHAR2(100) NOT NULL,

Descriere VARCHAR2(500) NOT NULL,

TipAlergie VARCHAR2(50) NOT NULL,

CodMedical VARCHAR2(20) NOT NULL,

-- Unique constraint

CONSTRAINT UQ\_ALERGIE\_COD\_MEDICAL UNIQUE (CodMedical)

);

CREATE TABLE MEDIC\_SPECIALIZARE (

IDMedic NUMBER(10),

IDSpecializare NUMBER(10),

DataObtinere DATE NOT NULL,

CertificatNr VARCHAR2(20) NOT NULL,

-- Primary key constraint (composite)

CONSTRAINT PK\_MEDIC\_SPECIALIZARE PRIMARY KEY (IDMedic, IDSpecializare),

-- Foreign key constraints

CONSTRAINT FK\_MEDIC\_SPECIALIZARE\_MEDIC FOREIGN KEY (IDMedic)

REFERENCES MEDIC(ID),

CONSTRAINT FK\_MEDIC\_SPECIALIZARE\_SPECIALIZARE FOREIGN KEY (IDSpecializare)

REFERENCES SPECIALIZARE(ID),

-- Unique constraint

CONSTRAINT UQ\_MEDIC\_SPECIALIZARE\_CERTIFICAT UNIQUE (CertificatNr)

);

CREATE TABLE PACIENT\_ALERGIE (

IDPacient NUMBER(10),

IDAlergie NUMBER(10),

DataDiagnostic DATE NOT NULL,

Observatii VARCHAR2(500),

Severitate NUMBER(1) NOT NULL,

-- Primary key constraint (composite)

CONSTRAINT PK\_PACIENT\_ALERGIE PRIMARY KEY (IDPacient, IDAlergie),

-- Foreign key constraints

CONSTRAINT FK\_PACIENT\_ALERGIE\_PACIENT FOREIGN KEY (IDPacient)

REFERENCES PACIENT(ID),

CONSTRAINT FK\_PACIENT\_ALERGIE\_ALERGIE FOREIGN KEY (IDAlergie)

REFERENCES ALERGIE(ID),

-- Check constraint

CONSTRAINT CHK\_PACIENT\_ALERGIE\_SEVERITATE CHECK (Severitate IN (1, 2, 3, 4, 5))

);

CREATE TABLE CONSULTATIE (

ID NUMBER(10) PRIMARY KEY,

IDProgramare NUMBER(10) NOT NULL,

DataConsultatie TIMESTAMP DEFAULT CURRENT\_TIMESTAMP NOT NULL,

Diagnostic VARCHAR2(500) NOT NULL,

Observatii VARCHAR2(500),

Recomandari VARCHAR2(500),

Urgenta NUMBER(1) NOT NULL,

-- Foreign key constraint

CONSTRAINT FK\_CONSULTATIE\_PROGRAMARE FOREIGN KEY (IDProgramare)

REFERENCES PROGRAMARE(ID),

-- Check constraint

CONSTRAINT CHK\_CONSULTATIE\_URGENTA CHECK (Urgenta IN (1, 2, 3, 4, 5))

);

CREATE TABLE RETETA (

ID NUMBER(10) PRIMARY KEY,

IDConsultatie NUMBER(10) NOT NULL,

DataPrescriere TIMESTAMP DEFAULT CURRENT\_TIMESTAMP NOT NULL,

DataExpirare TIMESTAMP NOT NULL,

Observatii VARCHAR2(500),

Status VARCHAR2(20) NOT NULL,

CodUnic VARCHAR2(20) NOT NULL,

-- Foreign key constraint

CONSTRAINT FK\_RETETA\_CONSULTATIE FOREIGN KEY (IDConsultatie)

REFERENCES CONSULTATIE(ID),

-- Unique constraint

CONSTRAINT UQ\_RETETA\_COD\_UNIC UNIQUE (CodUnic),

-- Check constraints

CONSTRAINT CHK\_RETETA\_DATA CHECK (DataExpirare > DataPrescriere),

CONSTRAINT CHK\_RETETA\_STATUS CHECK (Status IN ('In asteptare', 'In curs', 'Finalizata'))

);

CREATE TABLE MEDICAMENT (

ID NUMBER(10) PRIMARY KEY,

Denumire VARCHAR2(100) NOT NULL,

SubstantaActiva VARCHAR2(200) NOT NULL,

Concentratie VARCHAR2(50) NOT NULL,

FormaFarmaceutica VARCHAR2(50) NOT NULL,

Producator VARCHAR2(100) NOT NULL,

PretUnitar NUMBER(10,2) NOT NULL,

StocDisponibil NUMBER(10) NOT NULL,

NecesitaReteta NUMBER(1) NOT NULL,

-- Check constraints

CONSTRAINT CHK\_MEDICAMENT\_CONCENTRATIE CHECK (

REGEXP\_LIKE(Concentratie, '^[0-9][0-9].\*$')

),

CONSTRAINT CHK\_MEDICAMENT\_PRET CHECK (PretUnitar > 0),

CONSTRAINT CHK\_MEDICAMENT\_STOC CHECK (StocDisponibil >= 0),

CONSTRAINT CHK\_MEDICAMENT\_NECESITA\_RETETA CHECK (NecesitaReteta IN (0, 1))

);

CREATE TABLE RETETA\_MEDICAMENT (

IDReteta NUMBER(10),

IDMedicament NUMBER(10),

Cantitate NUMBER(10) NOT NULL,

Dozaj VARCHAR2(50) NOT NULL,

DurataTratament NUMBER(10) NOT NULL,

InstructiuniAdministrare VARCHAR2(500) NOT NULL,

-- Primary key constraint (composite)

CONSTRAINT PK\_RETETA\_MEDICAMENT PRIMARY KEY (IDReteta, IDMedicament),

-- Foreign key constraints

CONSTRAINT FK\_RETETA\_MEDICAMENT\_RETETA FOREIGN KEY (IDReteta)

REFERENCES RETETA(ID),

CONSTRAINT FK\_RETETA\_MEDICAMENT\_MEDICAMENT FOREIGN KEY (IDMedicament)

REFERENCES MEDICAMENT(ID),

-- Check constraints

CONSTRAINT CHK\_RETETA\_MEDICAMENT\_CANTITATE CHECK (Cantitate > 0),

CONSTRAINT CHK\_RETETA\_MEDICAMENT\_DURATA CHECK (DurataTratament > 0)

);

COMMIT;

**3️. Definire Secvențe (3\_secvente.sql)**

* Creare secvențe pentru generarea automată a ID-urilor
* Prefix-uri distincte pentru fiecare tip de entitate:
  + 1xxx... pentru PACIENT
  + 2xxx... pentru MEDIC
  + 3xxx... pentru DEPARTAMENT
  + etc.
* 🎯 Avantaj: Identificare ușoară a tipului de entitate după ID

SAVEPOINT start\_transaction;

CREATE SEQUENCE SEQ\_PACIENT\_ID

START WITH 1000001

INCREMENT BY 1

NOCACHE

NOCYCLE;

CREATE SEQUENCE SEQ\_MEDIC\_ID

START WITH 2000001

INCREMENT BY 1

NOCACHE

NOCYCLE;

CREATE SEQUENCE SEQ\_DEPARTAMENT\_ID

START WITH 3000001

INCREMENT BY 1

NOCACHE

NOCYCLE;

CREATE SEQUENCE SEQ\_PROGRAMARE\_ID

START WITH 4000001

INCREMENT BY 1

NOCACHE

NOCYCLE;

CREATE SEQUENCE SEQ\_CONSULTATIE\_ID

START WITH 5000001

INCREMENT BY 1

NOCACHE

NOCYCLE;

CREATE SEQUENCE SEQ\_RETETA\_ID

START WITH 6000001

INCREMENT BY 1

NOCACHE

NOCYCLE;

CREATE SEQUENCE SEQ\_MEDICAMENT\_ID

START WITH 7000001

INCREMENT BY 1

NOCACHE

NOCYCLE;

CREATE SEQUENCE SEQ\_ALERGIE\_ID

START WITH 8000001

INCREMENT BY 1

NOCACHE

NOCYCLE;

CREATE SEQUENCE SEQ\_SPECIALIZARE\_ID

START WITH 9000001

INCREMENT BY 1

NOCACHE

NOCYCLE;

COMMIT;

**4️. Creare triggere (4\_trigger.sql)**

* Creare trigger-uri pentru validare și control

SAVEPOINT start\_transaction;

CREATE OR REPLACE TRIGGER TRG\_PACIENT\_ID

BEFORE INSERT ON PACIENT

FOR EACH ROW

BEGIN

IF :NEW.ID IS NULL THEN

:NEW.ID := SEQ\_PACIENT\_ID.NEXTVAL;

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_MEDIC\_ID

BEFORE INSERT ON MEDIC

FOR EACH ROW

BEGIN

IF :NEW.ID IS NULL THEN

:NEW.ID := SEQ\_MEDIC\_ID.NEXTVAL;

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_PROGRAMARE\_ID

BEFORE INSERT ON PROGRAMARE

FOR EACH ROW

BEGIN

IF :NEW.ID IS NULL THEN

:NEW.ID := SEQ\_PROGRAMARE\_ID.NEXTVAL;

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_CONSULTATIE\_ID

BEFORE INSERT ON CONSULTATIE

FOR EACH ROW

BEGIN

IF :NEW.ID IS NULL THEN

:NEW.ID := SEQ\_CONSULTATIE\_ID.NEXTVAL;

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_RETETA\_ID

BEFORE INSERT ON RETETA

FOR EACH ROW

BEGIN

IF :NEW.ID IS NULL THEN

:NEW.ID := SEQ\_RETETA\_ID.NEXTVAL;

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_MEDICAMENT\_ID

BEFORE INSERT ON MEDICAMENT

FOR EACH ROW

BEGIN

IF :NEW.ID IS NULL THEN

:NEW.ID := SEQ\_MEDICAMENT\_ID.NEXTVAL;

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_ALERGIE\_ID

BEFORE INSERT ON ALERGIE

FOR EACH ROW

BEGIN

IF :NEW.ID IS NULL THEN

:NEW.ID := SEQ\_ALERGIE\_ID.NEXTVAL;

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_SPECIALIZARE\_ID

BEFORE INSERT ON SPECIALIZARE

FOR EACH ROW

BEGIN

IF :NEW.ID IS NULL THEN

:NEW.ID := SEQ\_SPECIALIZARE\_ID.NEXTVAL;

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_DEPARTAMENT\_ID

BEFORE INSERT ON DEPARTAMENT

FOR EACH ROW

BEGIN

IF :NEW.ID IS NULL THEN

:NEW.ID := SEQ\_DEPARTAMENT\_ID.NEXTVAL;

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_RETETA\_UNIQUE\_CONSULTATIE

BEFORE INSERT OR UPDATE ON RETETA

FOR EACH ROW

DECLARE

v\_count NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM RETETA

WHERE IDConsultatie = :NEW.IDConsultatie

AND (:NEW.ID IS NULL OR ID != :NEW.ID);

IF v\_count > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'O consultație poate avea o singură rețetă.');

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_CONSULTATIE\_UNIQUE\_PROGRAMARE

BEFORE INSERT OR UPDATE ON CONSULTATIE

FOR EACH ROW

DECLARE

v\_count NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM CONSULTATIE

WHERE IDProgramare = :NEW.IDProgramare

AND (:NEW.ID IS NULL OR ID != :NEW.ID);

IF v\_count > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'O programare poate avea o singură consultație.');

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_RETETA\_MEDICAMENT\_STOC

BEFORE INSERT OR UPDATE ON RETETA\_MEDICAMENT

FOR EACH ROW

DECLARE

v\_stoc\_disponibil NUMBER;

BEGIN

SELECT StocDisponibil

INTO v\_stoc\_disponibil

FROM MEDICAMENT

WHERE ID = :NEW.IDMedicament;

IF :NEW.Cantitate > v\_stoc\_disponibil THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Cantitatea solicitată depășește stocul disponibil.');

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_RETETA\_MEDICAMENT\_UPDATE\_STOC

AFTER INSERT OR UPDATE OR DELETE ON RETETA\_MEDICAMENT

FOR EACH ROW

BEGIN

IF INSERTING THEN

UPDATE MEDICAMENT

SET StocDisponibil = StocDisponibil - :NEW.Cantitate

WHERE ID = :NEW.IDMedicament;

ELSIF UPDATING THEN

UPDATE MEDICAMENT

SET StocDisponibil = StocDisponibil + :OLD.Cantitate - :NEW.Cantitate

WHERE ID = :NEW.IDMedicament;

ELSIF DELETING THEN

UPDATE MEDICAMENT

SET StocDisponibil = StocDisponibil + :OLD.Cantitate

WHERE ID = :OLD.IDMedicament;

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_PACIENT\_DATA\_NASTERII

BEFORE INSERT OR UPDATE OF DataNasterii ON PACIENT

FOR EACH ROW

BEGIN

IF :NEW.DataNasterii >= SYSDATE THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Data nașterii trebuie să fie în trecut.');

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_MEDIC\_DATA\_ANGAJARE

BEFORE INSERT OR UPDATE OF DataAngajare ON MEDIC

FOR EACH ROW

BEGIN

IF :NEW.DataAngajare > TRUNC(SYSDATE) THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Data angajării nu poate fi în viitor.');

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_PROGRAMARE\_DATA

BEFORE INSERT OR UPDATE OF DataProgramare ON PROGRAMARE

FOR EACH ROW

BEGIN

IF :NEW.DataProgramare < TRUNC(SYSDATE) THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Data programării nu poate fi în trecut.');

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_MEDIC\_SPECIALIZARE\_DATA

BEFORE INSERT OR UPDATE OF DataObtinere ON MEDIC\_SPECIALIZARE

FOR EACH ROW

BEGIN

IF :NEW.DataObtinere > TRUNC(SYSDATE) THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Data obținerii specializării nu poate fi în viitor.');

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_PACIENT\_ALERGIE\_DATA

BEFORE INSERT OR UPDATE OF DataDiagnostic ON PACIENT\_ALERGIE

FOR EACH ROW

BEGIN

IF :NEW.DataDiagnostic > TRUNC(SYSDATE) THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Data diagnosticului nu poate fi în viitor.');

END IF;

END;

CREATE OR REPLACE TRIGGER TRG\_CONSULTATIE\_DATA

BEFORE INSERT OR UPDATE OF DataConsultatie ON CONSULTATIE

FOR EACH ROW

BEGIN

IF TRUNC(:NEW.DataConsultatie) != TRUNC(SYSDATE) THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Data consultației trebuie să fie data curentă.');

END IF;

END;

COMMIT;

**5️. Inserare Date (5\_inregistrari.sql)**

Ordinea inserării trebuie să respecte dependențele:

1. Mai întâi tabelele independente
2. Apoi tabelele dependente, în ordinea dependențelor
3. La final tabelele asociative

**🔑 Aspecte Importante**

* Toate ID-urile sunt de tip NUMBER(10)
* Se folosește SAVEPOINT pentru control tranzacțional
* Constrângeri de integritate implementate prin:
  + Chei primare (PRIMARY KEY)
  + Chei externe (FOREIGN KEY)
  + Constrângeri CHECK pentru validări
  + Constrângeri UNIQUE pentru unicitate

SAVEPOINT start\_transaction;

-- DEPARTAMENT (fără IdSefDepartament inițial)

INSERT INTO DEPARTAMENT (NumeDepartament, Locatie, BugetAlocat, NrPaturi, TelefonContact) VALUES

('Cardiologie', 'Etaj 2, Aripa Est', 500000.00, 20, '0755123456');

INSERT INTO DEPARTAMENT (NumeDepartament, Locatie, BugetAlocat, NrPaturi, TelefonContact) VALUES

('Neurologie', 'Etaj 3, Aripa Vest', 450000.00, 15, '0755123457');

INSERT INTO DEPARTAMENT (NumeDepartament, Locatie, BugetAlocat, NrPaturi, TelefonContact) VALUES

('Pediatrie', 'Etaj 1, Aripa Sud', 400000.00, 25, '0755123458');

INSERT INTO DEPARTAMENT (NumeDepartament, Locatie, BugetAlocat, NrPaturi, TelefonContact) VALUES

('Chirurgie', 'Etaj 4, Aripa Nord', 600000.00, 30, '0755123459');

INSERT INTO DEPARTAMENT (NumeDepartament, Locatie, BugetAlocat, NrPaturi, TelefonContact) VALUES

('Ortopedie', 'Etaj 2, Aripa Vest', 350000.00, 18, '0755123460');

-- MEDIC

INSERT INTO MEDIC (Nume, Prenume, CNP, Telefon, Email, DataAngajare, GradProfesional, NrLicenta, IdDepartament) VALUES

('Popescu', 'Ion', '1780101123456', '0722123456', 'ion.popescu@spital.ro', TO\_DATE('2020-01-15', 'YYYY-MM-DD'), 'Primar', 'L123456', 3000001);

INSERT INTO MEDIC (Nume, Prenume, CNP, Telefon, Email, DataAngajare, GradProfesional, NrLicenta, IdDepartament) VALUES

('Ionescu', 'Maria', '2800202123456', '0722123457', 'maria.ionescu@spital.ro', TO\_DATE('2019-03-20', 'YYYY-MM-DD'), 'Specialist', 'L123457', 3000002);

INSERT INTO MEDIC (Nume, Prenume, CNP, Telefon, Email, DataAngajare, GradProfesional, NrLicenta, IdDepartament) VALUES

('Georgescu', 'Ana', '2810303123456', '0722123458', 'ana.georgescu@spital.ro', TO\_DATE('2021-06-10', 'YYYY-MM-DD'), 'Primar', 'L123458', 3000003);

INSERT INTO MEDIC (Nume, Prenume, CNP, Telefon, Email, DataAngajare, GradProfesional, NrLicenta, IdDepartament) VALUES

('Vasilescu', 'Dan', '1820404123456', '0722123459', 'dan.vasilescu@spital.ro', TO\_DATE('2018-09-01', 'YYYY-MM-DD'), 'Specialist', 'L123459', 3000004);

INSERT INTO MEDIC (Nume, Prenume, CNP, Telefon, Email, DataAngajare, GradProfesional, NrLicenta, IdDepartament) VALUES

('Marinescu', 'Elena', '2830505123456', '0722123460', 'elena.marinescu@spital.ro', TO\_DATE('2022-01-05', 'YYYY-MM-DD'), 'Rezident', 'L123460', 3000005);

-- Update DEPARTAMENT cu IdSefDepartament

UPDATE DEPARTAMENT SET IdSefDepartament = 2000001 WHERE ID = 3000001;

UPDATE DEPARTAMENT SET IdSefDepartament = 2000002 WHERE ID = 3000002;

UPDATE DEPARTAMENT SET IdSefDepartament = 2000003 WHERE ID = 3000003;

UPDATE DEPARTAMENT SET IdSefDepartament = 2000004 WHERE ID = 3000004;

UPDATE DEPARTAMENT SET IdSefDepartament = 2000005 WHERE ID = 3000005;

-- SPECIALIZARE

INSERT INTO SPECIALIZARE (Denumire, Descriere, Nivel, CodSpecializare) VALUES

('Cardiologie', 'Specializare în boli cardiovasculare', 'Avansat', 'CARD001');

INSERT INTO SPECIALIZARE (Denumire, Descriere, Nivel, CodSpecializare) VALUES

('Neurologie', 'Specializare în boli neurologice', 'Avansat', 'NEUR001');

INSERT INTO SPECIALIZARE (Denumire, Descriere, Nivel, CodSpecializare) VALUES

('Pediatrie', 'Specializare în îngrijirea copiilor', 'Mediu', 'PED001');

INSERT INTO SPECIALIZARE (Denumire, Descriere, Nivel, CodSpecializare) VALUES

('Chirurgie Generală', 'Specializare în intervenții chirurgicale', 'Avansat', 'CHIR001');

INSERT INTO SPECIALIZARE (Denumire, Descriere, Nivel, CodSpecializare) VALUES

('Ortopedie', 'Specializare în afecțiuni ale sistemului osteoarticular', 'Mediu', 'ORT001');

-- PACIENT

INSERT INTO PACIENT (Nume, Prenume, CNP, Adresa, Telefon, Email, DataNasterii, Sex, GrupaSanguina) VALUES

('Popa', 'Alexandru', '1900101123456', 'Str. Primaverii 10, București', '0733123456', 'alex.popa@email.com', TO\_DATE('1990-01-01', 'YYYY-MM-DD'), 'M', 'A+');

INSERT INTO PACIENT (Nume, Prenume, CNP, Adresa, Telefon, Email, DataNasterii, Sex, GrupaSanguina) VALUES

('Dumitrescu', 'Elena', '2910202123456', 'Str. Florilor 20, București', '0733123457', 'elena.dumitrescu@email.com', TO\_DATE('1991-02-02', 'YYYY-MM-DD'), 'F', 'O+');

INSERT INTO PACIENT (Nume, Prenume, CNP, Adresa, Telefon, Email, DataNasterii, Sex, GrupaSanguina) VALUES

('Stanescu', 'Mihai', '1920303123456', 'Str. Victoriei 30, București', '0733123458', 'mihai.stanescu@email.com', TO\_DATE('1992-03-03', 'YYYY-MM-DD'), 'M', 'B+');

INSERT INTO PACIENT (Nume, Prenume, CNP, Adresa, Telefon, Email, DataNasterii, Sex, GrupaSanguina) VALUES

('Constantinescu', 'Maria', '2930404123456', 'Str. Unirii 40, București', '0733123459', 'maria.const@email.com', TO\_DATE('1993-04-04', 'YYYY-MM-DD'), 'F', 'AB+');

INSERT INTO PACIENT (Nume, Prenume, CNP, Adresa, Telefon, Email, DataNasterii, Sex, GrupaSanguina) VALUES

('Diaconu', 'Andrei', '1940505123456', 'Str. Libertatii 50, București', '0733123460', 'andrei.diaconu@email.com', TO\_DATE('1994-05-05', 'YYYY-MM-DD'), 'M', 'A-');

-- ALERGIE

INSERT INTO ALERGIE (Denumire, Descriere, TipAlergie, CodMedical) VALUES

('Penicilină', 'Alergie la antibiotice din grupa penicilinei', 'Medicamentoasă', 'AL001');

INSERT INTO ALERGIE (Denumire, Descriere, TipAlergie, CodMedical) VALUES

('Polen', 'Alergie sezonieră la polen', 'Sezonieră', 'AL002');

INSERT INTO ALERGIE (Denumire, Descriere, TipAlergie, CodMedical) VALUES

('Lactoza', 'Intoleranță la lactoză', 'Alimentară', 'AL003');

INSERT INTO ALERGIE (Denumire, Descriere, TipAlergie, CodMedical) VALUES

('Acarieni', 'Alergie la acarieni din praf', 'Respiratorie', 'AL004');

INSERT INTO ALERGIE (Denumire, Descriere, TipAlergie, CodMedical) VALUES

('Latex', 'Alergie la latex', 'Contact', 'AL005');

-- MEDICAMENT

INSERT INTO MEDICAMENT (Denumire, SubstantaActiva, Concentratie, FormaFarmaceutica, Producator, PretUnitar, StocDisponibil, NecesitaReteta) VALUES

('Augmentin', 'Amoxicilină', '875mg', 'Comprimate', 'GSK', 35.50, 100, 1);

INSERT INTO MEDICAMENT (Denumire, SubstantaActiva, Concentratie, FormaFarmaceutica, Producator, PretUnitar, StocDisponibil, NecesitaReteta) VALUES

('Nurofen', 'Ibuprofen', '400mg', 'Comprimate', 'Reckitt', 15.75, 200, 0);

INSERT INTO MEDICAMENT (Denumire, SubstantaActiva, Concentratie, FormaFarmaceutica, Producator, PretUnitar, StocDisponibil, NecesitaReteta) VALUES

('Paracetamol', 'Paracetamol', '500mg', 'Comprimate', 'Zentiva', 12.30, 300, 0);

INSERT INTO MEDICAMENT (Denumire, SubstantaActiva, Concentratie, FormaFarmaceutica, Producator, PretUnitar, StocDisponibil, NecesitaReteta) VALUES

('Ventolin', 'Salbutamol', '100mcg', 'Inhaler', 'GSK', 45.00, 50, 1);

INSERT INTO MEDICAMENT (Denumire, SubstantaActiva, Concentratie, FormaFarmaceutica, Producator, PretUnitar, StocDisponibil, NecesitaReteta) VALUES

('Omeprazol', 'Omeprazol', '20mg', 'Capsule', 'Terapia', 25.80, 150, 1);

-- PROGRAMARE

INSERT INTO PROGRAMARE (IDPacient, IDMedic, DataProgramare, OraProgramare, Status, MotivPrezentare, Observatii) VALUES

(1000001, 2000001, TRUNC(SYSDATE), '09:00', 'Programata', 'Consult cardiologic', 'Prima vizită');

INSERT INTO PROGRAMARE (IDPacient, IDMedic, DataProgramare, OraProgramare, Status, MotivPrezentare, Observatii) VALUES

(1000002, 2000002, TRUNC(SYSDATE), '10:30', 'Programata', 'Dureri de cap', 'Pacient cu antecedente');

INSERT INTO PROGRAMARE (IDPacient, IDMedic, DataProgramare, OraProgramare, Status, MotivPrezentare, Observatii) VALUES

(1000003, 2000003, TRUNC(SYSDATE), '11:45', 'Programata', 'Control periodic', NULL);

INSERT INTO PROGRAMARE (IDPacient, IDMedic, DataProgramare, OraProgramare, Status, MotivPrezentare, Observatii) VALUES

(1000004, 2000004, TRUNC(SYSDATE), '13:15', 'Programata', 'Evaluare preoperatorie', 'Urgent');

INSERT INTO PROGRAMARE (IDPacient, IDMedic, DataProgramare, OraProgramare, Status, MotivPrezentare, Observatii) VALUES

(1000005, 2000005, TRUNC(SYSDATE), '14:30', 'Programata', 'Dureri articulare', NULL);

INSERT INTO PROGRAMARE (IDPacient, IDMedic, DataProgramare, OraProgramare, Status, MotivPrezentare, Observatii) VALUES

(1000001, 2000002, TRUNC(SYSDATE+1), '09:15', 'Programata', 'Second opinion', NULL);

INSERT INTO PROGRAMARE (IDPacient, IDMedic, DataProgramare, OraProgramare, Status, MotivPrezentare, Observatii) VALUES

(1000002, 2000003, TRUNC(SYSDATE+1), '10:45', 'Programata', 'Evaluare rezultate', NULL);

INSERT INTO PROGRAMARE (IDPacient, IDMedic, DataProgramare, OraProgramare, Status, MotivPrezentare, Observatii) VALUES

(1000003, 2000004, TRUNC(SYSDATE+1), '12:00', 'Programata', 'Control postoperator', 'Follow-up');

INSERT INTO PROGRAMARE (IDPacient, IDMedic, DataProgramare, OraProgramare, Status, MotivPrezentare, Observatii) VALUES

(1000004, 2000005, TRUNC(SYSDATE+1), '13:30', 'Programata', 'Consultație control', NULL);

INSERT INTO PROGRAMARE (IDPacient, IDMedic, DataProgramare, OraProgramare, Status, MotivPrezentare, Observatii) VALUES

(1000005, 2000001, TRUNC(SYSDATE+1), '15:00', 'Programata', 'Evaluare periodică', NULL);

-- CONSULTATIE (pentru programările din ziua curentă)

INSERT INTO CONSULTATIE (IDProgramare, Diagnostic, Observatii, Recomandari, Urgenta) VALUES

(4000001, 'Hipertensiune arterială', 'Tensiune 150/90', 'Regim alimentar strict', 3);

INSERT INTO CONSULTATIE (IDProgramare, Diagnostic, Observatii, Recomandari, Urgenta) VALUES

(4000002, 'Migrenă cronică', 'Dureri frecvente', 'Evitarea factorilor declanșatori', 2);

INSERT INTO CONSULTATIE (IDProgramare, Diagnostic, Observatii, Recomandari, Urgenta) VALUES

(4000003, 'Stare generală bună', 'Control de rutină', 'Menținerea stilului de viață actual', 1);

INSERT INTO CONSULTATIE (IDProgramare, Diagnostic, Observatii, Recomandari, Urgenta) VALUES

(4000004, 'Hernie de disc', 'Necesită intervenție', 'Pregătire preoperatorie', 4);

INSERT INTO CONSULTATIE (IDProgramare, Diagnostic, Observatii, Recomandari, Urgenta) VALUES

(4000005, 'Artroză genunchi', 'Dureri la mers', 'Fizioterapie', 2);

-- RETETA (pentru consultațiile efectuate)

INSERT INTO RETETA (IDConsultatie, DataExpirare, Observatii, Status, CodUnic) VALUES

(5000001, SYSDATE + 30, 'A se administra dimineața', 'In asteptare', 'RX001');

INSERT INTO RETETA (IDConsultatie, DataExpirare, Observatii, Status, CodUnic) VALUES

(5000002, SYSDATE + 30, 'A se administra la nevoie', 'In asteptare', 'RX002');

INSERT INTO RETETA (IDConsultatie, DataExpirare, Observatii, Status, CodUnic) VALUES

(5000003, SYSDATE + 30, NULL, 'In asteptare', 'RX003');

INSERT INTO RETETA (IDConsultatie, DataExpirare, Observatii, Status, CodUnic) VALUES

(5000004, SYSDATE + 30, 'Tratament preoperator', 'In asteptare', 'RX004');

INSERT INTO RETETA (IDConsultatie, DataExpirare, Observatii, Status, CodUnic) VALUES

(5000005, SYSDATE + 30, 'Tratament antiinflamator', 'In asteptare', 'RX005');

-- Tabele asociative (10 înregistrări pentru fiecare)

-- MEDIC\_SPECIALIZARE

INSERT INTO MEDIC\_SPECIALIZARE (IDMedic, IDSpecializare, DataObtinere, CertificatNr) VALUES

(2000001, 9000001, TO\_DATE('2019-01-15', 'YYYY-MM-DD'), 'CERT001');

INSERT INTO MEDIC\_SPECIALIZARE (IDMedic, IDSpecializare, DataObtinere, CertificatNr) VALUES

(2000001, 9000002, TO\_DATE('2019-06-20', 'YYYY-MM-DD'), 'CERT002');

INSERT INTO MEDIC\_SPECIALIZARE (IDMedic, IDSpecializare, DataObtinere, CertificatNr) VALUES

(2000002, 9000002, TO\_DATE('2018-03-10', 'YYYY-MM-DD'), 'CERT003');

INSERT INTO MEDIC\_SPECIALIZARE (IDMedic, IDSpecializare, DataObtinere, CertificatNr) VALUES

(2000002, 9000003, TO\_DATE('2018-09-15', 'YYYY-MM-DD'), 'CERT004');

INSERT INTO MEDIC\_SPECIALIZARE (IDMedic, IDSpecializare, DataObtinere, CertificatNr) VALUES

(2000003, 9000003, TO\_DATE('2020-02-28', 'YYYY-MM-DD'), 'CERT005');

INSERT INTO MEDIC\_SPECIALIZARE (IDMedic, IDSpecializare, DataObtinere, CertificatNr) VALUES

(2000003, 9000004, TO\_DATE('2020-08-15', 'YYYY-MM-DD'), 'CERT006');

INSERT INTO MEDIC\_SPECIALIZARE (IDMedic, IDSpecializare, DataObtinere, CertificatNr) VALUES

(2000004, 9000004, TO\_DATE('2017-11-20', 'YYYY-MM-DD'), 'CERT007');

INSERT INTO MEDIC\_SPECIALIZARE (IDMedic, IDSpecializare, DataObtinere, CertificatNr) VALUES

(2000004, 9000005, TO\_DATE('2018-04-25', 'YYYY-MM-DD'), 'CERT008');

INSERT INTO MEDIC\_SPECIALIZARE (IDMedic, IDSpecializare, DataObtinere, CertificatNr) VALUES

(2000005, 9000005, TO\_DATE('2021-07-30', 'YYYY-MM-DD'), 'CERT009');

INSERT INTO MEDIC\_SPECIALIZARE (IDMedic, IDSpecializare, DataObtinere, CertificatNr) VALUES

(2000005, 9000001, TO\_DATE('2021-12-15', 'YYYY-MM-DD'), 'CERT010');

-- PACIENT\_ALERGIE

INSERT INTO PACIENT\_ALERGIE (IDPacient, IDAlergie, DataDiagnostic, Observatii, Severitate) VALUES

(1000001, 8000001, TO\_DATE('2020-01-15', 'YYYY-MM-DD'), 'Reacție severă la penicilină', 5);

INSERT INTO PACIENT\_ALERGIE (IDPacient, IDAlergie, DataDiagnostic, Observatii, Severitate) VALUES

(1000001, 8000002, TO\_DATE('2020-03-20', 'YYYY-MM-DD'), 'Rinită alergică sezonieră', 3);

INSERT INTO PACIENT\_ALERGIE (IDPacient, IDAlergie, DataDiagnostic, Observatii, Severitate) VALUES

(1000002, 8000002, TO\_DATE('2019-05-10', 'YYYY-MM-DD'), 'Manifestări moderate', 2);

INSERT INTO PACIENT\_ALERGIE (IDPacient, IDAlergie, DataDiagnostic, Observatii, Severitate) VALUES

(1000002, 8000003, TO\_DATE('2019-08-15', 'YYYY-MM-DD'), 'Intoleranță confirmată', 4);

INSERT INTO PACIENT\_ALERGIE (IDPacient, IDAlergie, DataDiagnostic, Observatii, Severitate) VALUES

(1000003, 8000003, TO\_DATE('2021-02-28', 'YYYY-MM-DD'), 'Simptome ușoare', 1);

INSERT INTO PACIENT\_ALERGIE (IDPacient, IDAlergie, DataDiagnostic, Observatii, Severitate) VALUES

(1000003, 8000004, TO\_DATE('2021-06-15', 'YYYY-MM-DD'), 'Necesită tratament continuu', 3);

INSERT INTO PACIENT\_ALERGIE (IDPacient, IDAlergie, DataDiagnostic, Observatii, Severitate) VALUES

(1000004, 8000004, TO\_DATE('2018-11-20', 'YYYY-MM-DD'), 'Agravare în sezonul rece', 4);

INSERT INTO PACIENT\_ALERGIE (IDPacient, IDAlergie, DataDiagnostic, Observatii, Severitate) VALUES

(1000004, 8000005, TO\_DATE('2019-04-25', 'YYYY-MM-DD'), 'Reacție moderată', 2);

INSERT INTO PACIENT\_ALERGIE (IDPacient, IDAlergie, DataDiagnostic, Observatii, Severitate) VALUES

(1000005, 8000005, TO\_DATE('2022-07-30', 'YYYY-MM-DD'), 'Sub observație', 3);

INSERT INTO PACIENT\_ALERGIE (IDPacient, IDAlergie, DataDiagnostic, Observatii, Severitate) VALUES

(1000005, 8000001, TO\_DATE('2022-12-15', 'YYYY-MM-DD'), 'Necesită atenție sporită', 4);

-- RETETA\_MEDICAMENT

INSERT INTO RETETA\_MEDICAMENT (IDReteta, IDMedicament, Cantitate, Dozaj, DurataTratament, InstructiuniAdministrare) VALUES

(6000001, 7000001, 1, '1cp x 2/zi', 10, 'Dimineața și seara, după masă');

INSERT INTO RETETA\_MEDICAMENT (IDReteta, IDMedicament, Cantitate, Dozaj, DurataTratament, InstructiuniAdministrare) VALUES

(6000001, 7000002, 2, '1cp la 8 ore', 7, 'La nevoie, maxim 3 pe zi');

INSERT INTO RETETA\_MEDICAMENT (IDReteta, IDMedicament, Cantitate, Dozaj, DurataTratament, InstructiuniAdministrare) VALUES

(6000002, 7000003, 3, '1cp la 6 ore', 5, 'În caz de durere');

INSERT INTO RETETA\_MEDICAMENT (IDReteta, IDMedicament, Cantitate, Dozaj, DurataTratament, InstructiuniAdministrare) VALUES

(6000002, 7000004, 1, '2 pufuri la nevoie', 30, 'În caz de criză');

INSERT INTO RETETA\_MEDICAMENT (IDReteta, IDMedicament, Cantitate, Dozaj, DurataTratament, InstructiuniAdministrare) VALUES

(6000003, 7000005, 2, '1cp/zi', 14, 'Dimineața, înainte de masă');

INSERT INTO RETETA\_MEDICAMENT (IDReteta, IDMedicament, Cantitate, Dozaj, DurataTratament, InstructiuniAdministrare) VALUES

(6000003, 7000001, 1, '1cp x 2/zi', 7, 'Dimineața și seara');

INSERT INTO RETETA\_MEDICAMENT (IDReteta, IDMedicament, Cantitate, Dozaj, DurataTratament, InstructiuniAdministrare) VALUES

(6000004, 7000002, 3, '1cp la 12 ore', 10, 'La durere');

INSERT INTO RETETA\_MEDICAMENT (IDReteta, IDMedicament, Cantitate, Dozaj, DurataTratament, InstructiuniAdministrare) VALUES

(6000004, 7000003, 2, '1cp la 8 ore', 5, 'După masă');

INSERT INTO RETETA\_MEDICAMENT (IDReteta, IDMedicament, Cantitate, Dozaj, DurataTratament, InstructiuniAdministrare) VALUES

(6000005, 7000004, 1, '2 pufuri de 3 ori/zi', 30, 'La nevoie');

INSERT INTO RETETA\_MEDICAMENT (IDReteta, IDMedicament, Cantitate, Dozaj, DurataTratament, InstructiuniAdministrare) VALUES

(6000005, 7000005, 1, '1cp seara', 14, 'Înainte de culcare');

COMMIT;

**6️. Drop (6\_drop.sql)**

* Drop toate obiectele create

-- 1. Drop trigger

DROP TRIGGER trg\_monitorizare\_schema;

-- 2. Drop tabelul MESAJE

DROP TABLE MESAJE;

-- 3. Drop package

DROP PACKAGE management\_spital;

-- 4. Drop restul trigger-urilor

DROP TRIGGER TRG\_PACIENT\_ID;

DROP TRIGGER TRG\_MEDIC\_ID;

DROP TRIGGER TRG\_PROGRAMARE\_ID;

DROP TRIGGER TRG\_CONSULTATIE\_ID;

DROP TRIGGER TRG\_RETETA\_ID;

DROP TRIGGER TRG\_MEDICAMENT\_ID;

DROP TRIGGER TRG\_ALERGIE\_ID;

DROP TRIGGER TRG\_SPECIALIZARE\_ID;

DROP TRIGGER TRG\_DEPARTAMENT\_ID;

DROP TRIGGER TRG\_RETETA\_UNIQUE\_CONSULTATIE;

DROP TRIGGER TRG\_CONSULTATIE\_UNIQUE\_PROGRAMARE;

DROP TRIGGER TRG\_RETETA\_MEDICAMENT\_STOC;

DROP TRIGGER TRG\_RETETA\_MEDICAMENT\_UPDATE\_STOC;

DROP TRIGGER TRG\_PACIENT\_DATA\_NASTERII;

DROP TRIGGER TRG\_MEDIC\_DATA\_ANGAJARE;

DROP TRIGGER TRG\_PROGRAMARE\_DATA;

DROP TRIGGER TRG\_MEDIC\_SPECIALIZARE\_DATA;

DROP TRIGGER TRG\_PACIENT\_ALERGIE\_DATA;

DROP TRIGGER TRG\_CONSULTATIE\_DATA;

-- 5. Drop tabele asociative

DROP TABLE RETETA\_MEDICAMENT;

DROP TABLE PACIENT\_ALERGIE;

DROP TABLE MEDIC\_SPECIALIZARE;

-- 6. Drop tabele cu foreign keys

DROP TABLE RETETA;

DROP TABLE CONSULTATIE;

DROP TABLE PROGRAMARE;

-- 7. Drop tabele principale

ALTER TABLE DEPARTAMENT DROP CONSTRAINT FK\_DEPARTAMENT\_SEF;

DROP TABLE MEDIC;

DROP TABLE DEPARTAMENT;

DROP TABLE PACIENT;

DROP TABLE SPECIALIZARE;

DROP TABLE ALERGIE;

DROP TABLE MEDICAMENT;

-- 8. Drop secvente

DROP SEQUENCE SEQ\_PACIENT\_ID;

DROP SEQUENCE SEQ\_MEDIC\_ID;

DROP SEQUENCE SEQ\_DEPARTAMENT\_ID;

DROP SEQUENCE SEQ\_PROGRAMARE\_ID;

DROP SEQUENCE SEQ\_CONSULTATIE\_ID;

DROP SEQUENCE SEQ\_RETETA\_ID;

DROP SEQUENCE SEQ\_MEDICAMENT\_ID;

DROP SEQUENCE SEQ\_ALERGIE\_ID;

DROP SEQUENCE SEQ\_SPECIALIZARE\_ID;

DROP SEQUENCE SEQ\_MESAJE;

## 7. Interogari

1. **Interogare care analizează performanța medicilor și departamentelor, combinând date despre consultații, pacienți și rețete. Folosește CTE-uri multiple, JOIN-uri (INNER, LEFT), funcții de agregare (COUNT, AVG, SUM), funcții analitice (RANK, DENSE\_RANK), funcții pentru date (MONTHS\_BETWEEN), CASE, subinterogări și GROUP BY/HAVING pentru a genera un raport comprehensiv despre activitatea medicală.**

WITH DepartamentStats AS (

SELECT

d.NumeDepartament,

COUNT(DISTINCT m.ID) AS NumarMedici,

COUNT(DISTINCT p.ID) AS NumarProgramari,

SUM(d.BugetAlocat) AS BugetTotal,

AVG(c.Urgenta) AS UrgentaMedie

FROM DEPARTAMENT d

LEFT JOIN MEDIC m ON d.ID = m.IDDepartament

LEFT JOIN PROGRAMARE p ON m.ID = p.IDMedic

LEFT JOIN CONSULTATIE c ON p.ID = c.IDProgramare

GROUP BY d.NumeDepartament

)

SELECT

ds.\*,

RANK() OVER (ORDER BY NumarProgramari DESC) AS RankActivitate,

CASE

WHEN BugetTotal > 500000 THEN 'Buget Mare'

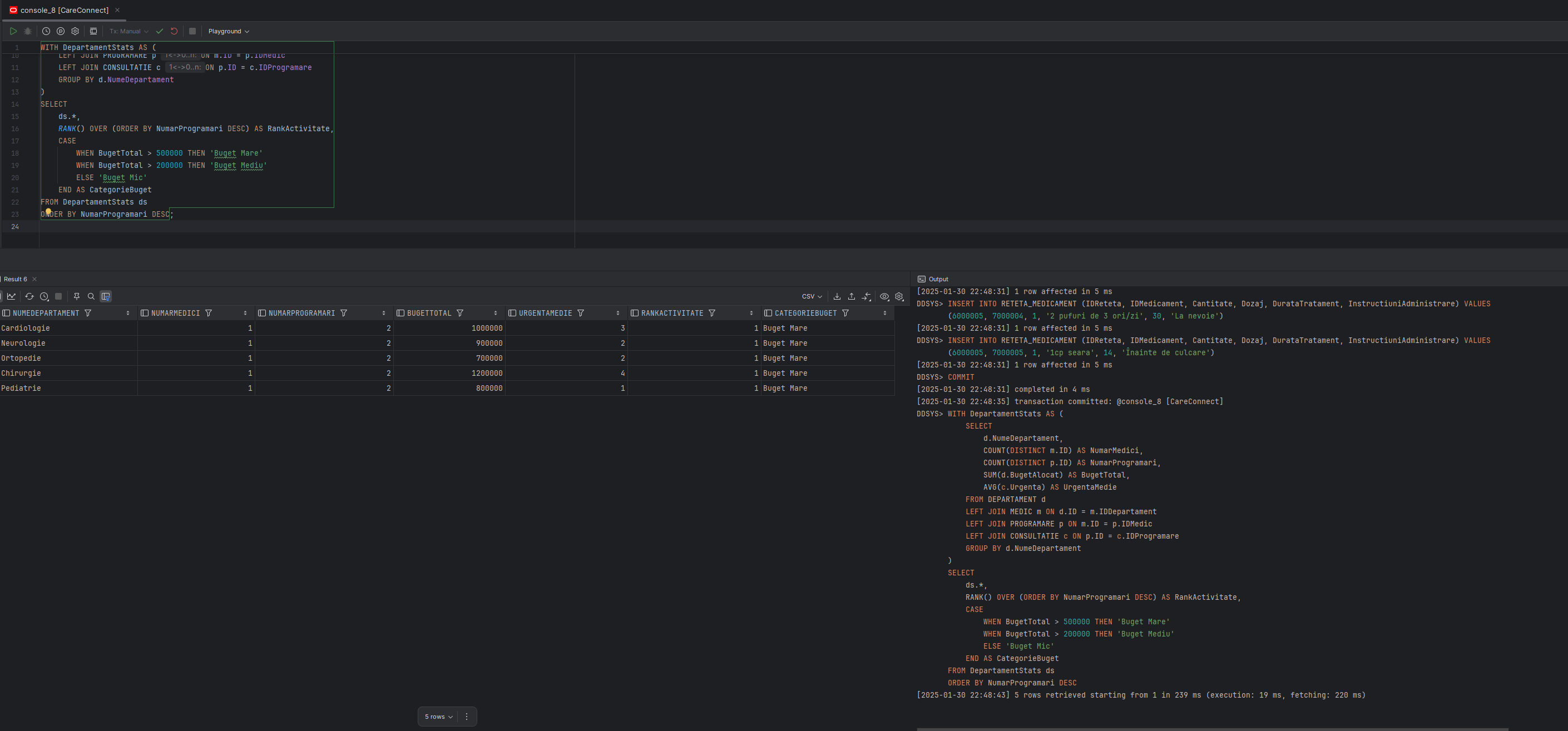
WHEN BugetTotal > 200000 THEN 'Buget Mediu'

ELSE 'Buget Mic'

END AS CategorieBuget

FROM DepartamentStats ds

ORDER BY NumarProgramari DESC;



1. **Interogare care generează un raport despre pacienți, incluzând vârsta, alergiile și istoricul consultațiilor. Utilizează funcții de date (MONTHS\_BETWEEN), funcții de agregare (COUNT), LISTAGG pentru concatenarea alergiilor, JOIN-uri multiple (LEFT JOIN) și GROUP BY cu HAVING pentru filtrarea pacienților cu minimum 3 programări. Oferă o vedere de ansamblu asupra pacienților frecvenți și stării lor medicale.**

SELECT

m.Nume || ' ' || m.Prenume AS NumeComplet,

LISTAGG(s.Denumire, ', ') WITHIN GROUP (ORDER BY s.Denumire) AS Specializari,

COUNT(DISTINCT p.ID) AS NumarProgramari,

TO\_CHAR(MIN(m.DataAngajare), 'DD-MON-YYYY') AS DataAngajare,

MONTHS\_BETWEEN(SYSDATE, m.DataAngajare)/12 AS AniVechime

FROM MEDIC m

LEFT JOIN MEDIC\_SPECIALIZARE ms ON m.ID = ms.IDMedic

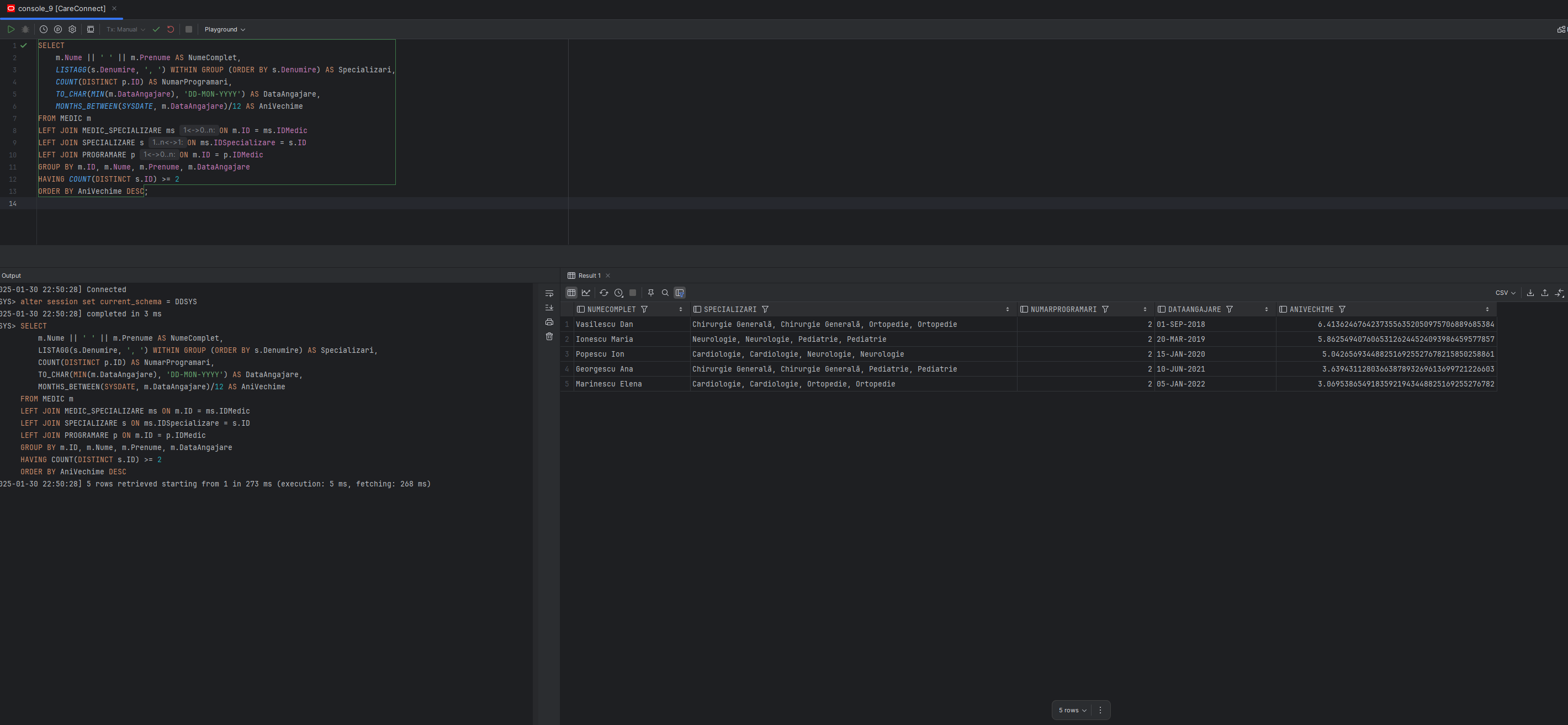
LEFT JOIN SPECIALIZARE s ON ms.IDSpecializare = s.ID

LEFT JOIN PROGRAMARE p ON m.ID = p.IDMedic

GROUP BY m.ID, m.Nume, m.Prenume, m.DataAngajare

HAVING COUNT(DISTINCT s.ID) >= 2

ORDER BY AniVechime DESC;



1. **Interogare care analizează statisticile despre medicamente și modul în care sunt prescrise, incluzând numărul de prescrieri, cantitățile și durata medie a tratamentelor. Folosește CTE pentru organizarea datelor, JOIN-uri multiple pentru conectarea informațiilor despre rețete și consultații, funcții de agregare (COUNT, SUM, AVG) pentru statistici, și funcții analitice (DENSE\_RANK) pentru clasificarea popularității medicamentelor. Rezultatele sunt ordonate după numărul de prescrieri, oferind o vedere de ansamblu asupra utilizării medicamentelor în clinică.**

SELECT p.Nume, p.Prenume

FROM PACIENT p

WHERE NOT EXISTS (

SELECT a.ID

FROM ALERGIE a

WHERE a.TipAlergie = 'Alimentară'

AND NOT EXISTS (

SELECT 1

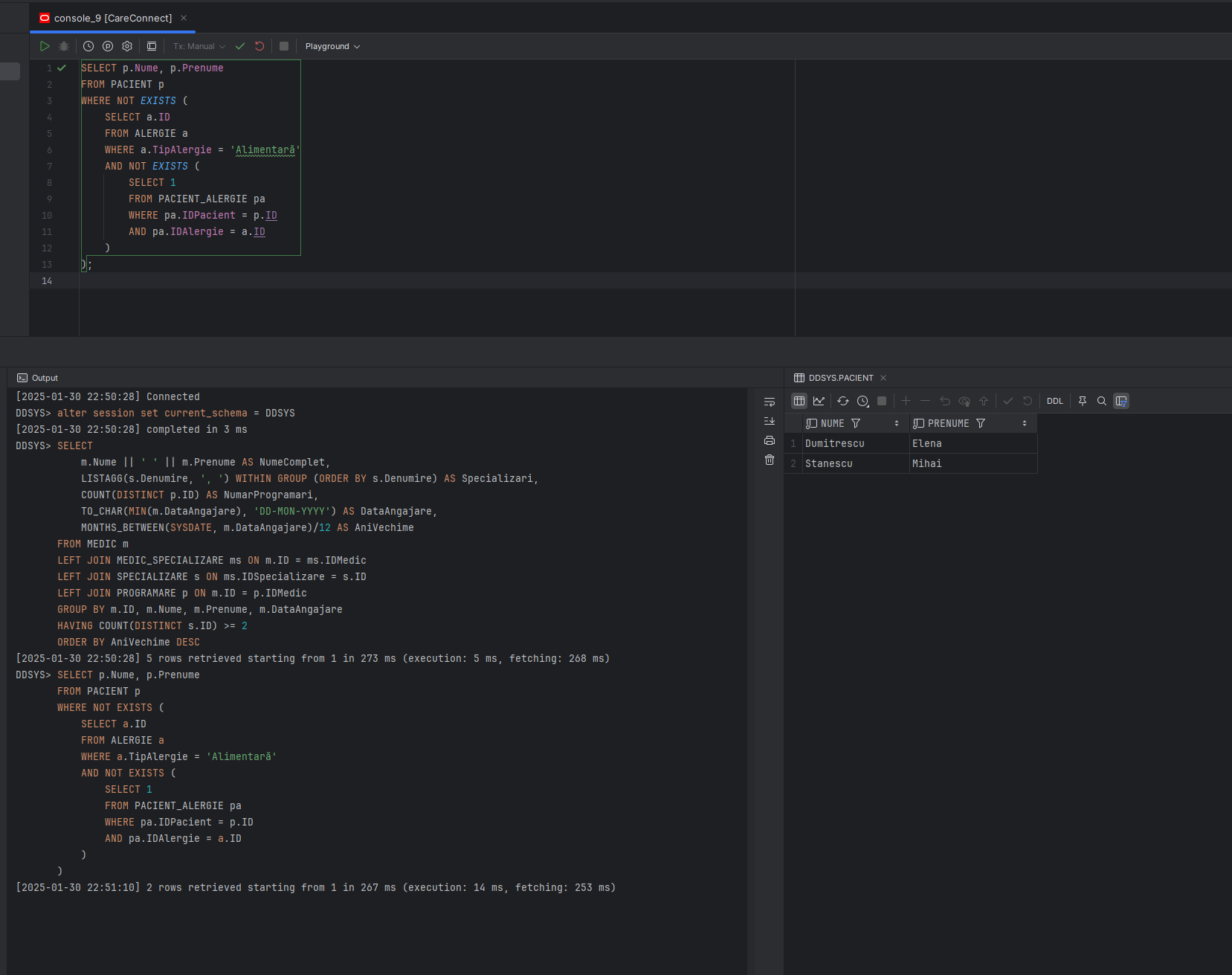
FROM PACIENT\_ALERGIE pa

WHERE pa.IDPacient = p.ID

AND pa.IDAlergie = a.ID

)

);



1. **Interogare care analizează programările medicale pe zile și ore, calculând rata de prezență a pacienților și urgența medie. Folosește funcții de dată (TO\_CHAR), agregare (COUNT, AVG), JOIN-uri multiple, GROUP BY cu HAVING, și CASE pentru ordonarea zilelor săptămânii. Include filtrare temporală (ADD\_MONTHS) și calculează statistici despre medici și departamente implicate în programări.**

SELECT

TO\_CHAR(DataProgramare, 'DY') AS ZiSaptamana,

CASE

WHEN TO\_NUMBER(SUBSTR(OraProgramare, 1, 2)) BETWEEN 8 AND 12 THEN 'Dimineața'

WHEN TO\_NUMBER(SUBSTR(OraProgramare, 1, 2)) BETWEEN 13 AND 16 THEN 'După-amiază'

ELSE 'Seara'

END AS IntervalOrar,

COUNT(\*) AS NumarProgramari,

ROUND(COUNT(\*) \* 100.0 / SUM(COUNT(\*)) OVER (), 2) AS ProcentDinTotal

FROM PROGRAMARE

GROUP BY

TO\_CHAR(DataProgramare, 'DY'),

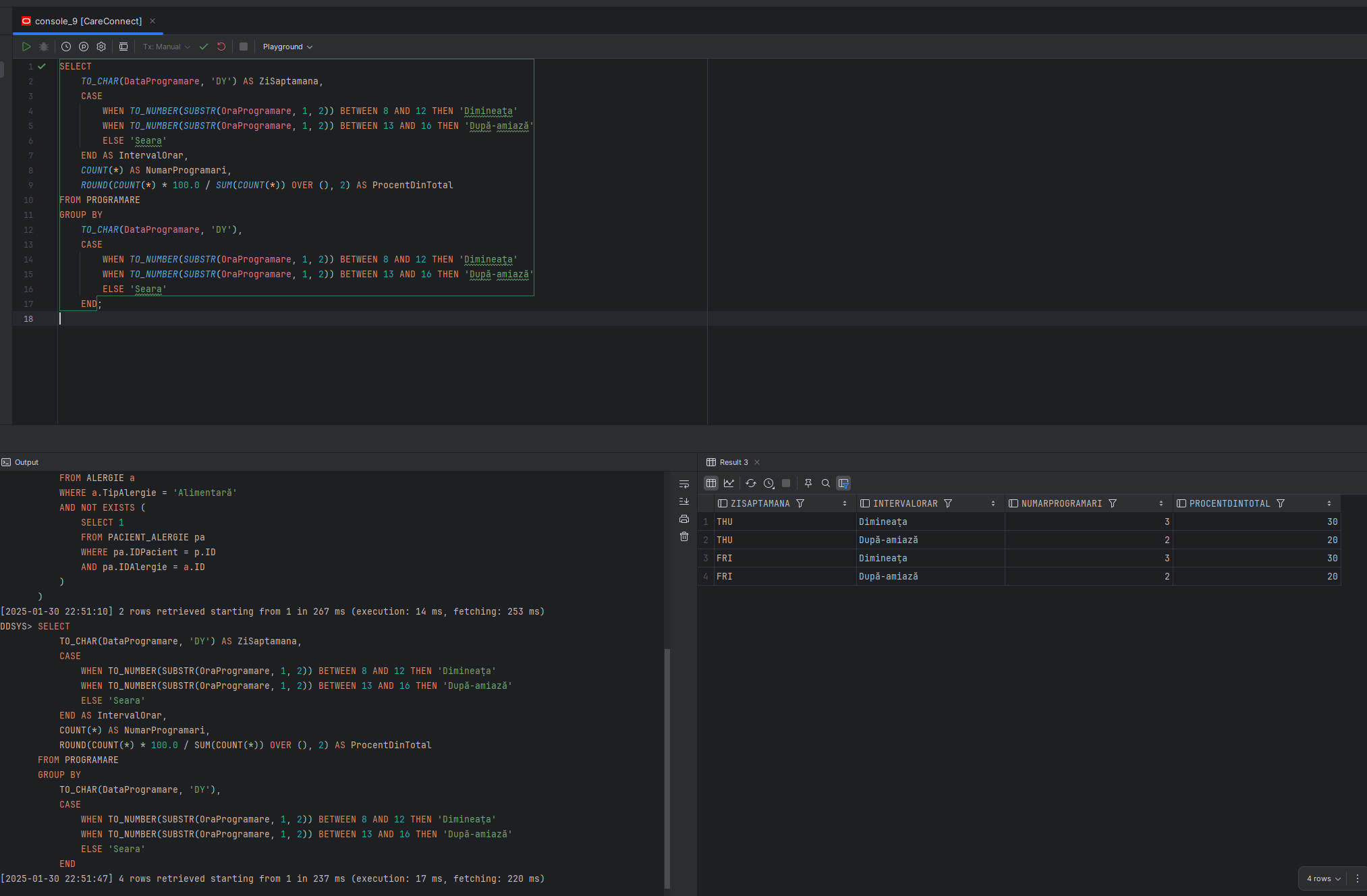
CASE

WHEN TO\_NUMBER(SUBSTR(OraProgramare, 1, 2)) BETWEEN 8 AND 12 THEN 'Dimineața'

WHEN TO\_NUMBER(SUBSTR(OraProgramare, 1, 2)) BETWEEN 13 AND 16 THEN 'După-amiază'

ELSE 'Seara'

END;



1. **Interogarea analizează eficiența departamentelor prin măsurarea ratei de finalizare a programărilor, costurilor per consultație și rețetă, folosind un CTE pentru agregarea datelor inițiale. Utilizează JOIN-uri multiple, funcții de agregare (COUNT, SUM), calcule procentuale, RANK pentru clasificare și NULLIF pentru evitarea împărțirii la zero. Rezultatul oferă o vedere de ansamblu asupra performanței fiecărui departament, ordonată după rata de finalizare.**

WITH MediciDepartament AS (

SELECT

d.NumeDepartament,

d.BugetAlocat,

m.Nume || ' ' || m.Prenume AS NumeMedic,

m.GradProfesional,

COUNT(p.ID) AS NumarProgramari,

ROW\_NUMBER() OVER (PARTITION BY d.ID ORDER BY COUNT(p.IDProgramare) DESC) AS RankInDepartament

FROM DEPARTAMENT d

LEFT JOIN MEDIC m ON d.ID = m.ID

LEFT JOIN PROGRAMARE p ON m.ID = p.IDMedic

GROUP BY d.NumeDepartament, d.BugetAlocat, m.Nume, m.Prenume, m.GradProfesional, d.ID

)

SELECT

NumeDepartament,

BugetAlocat,

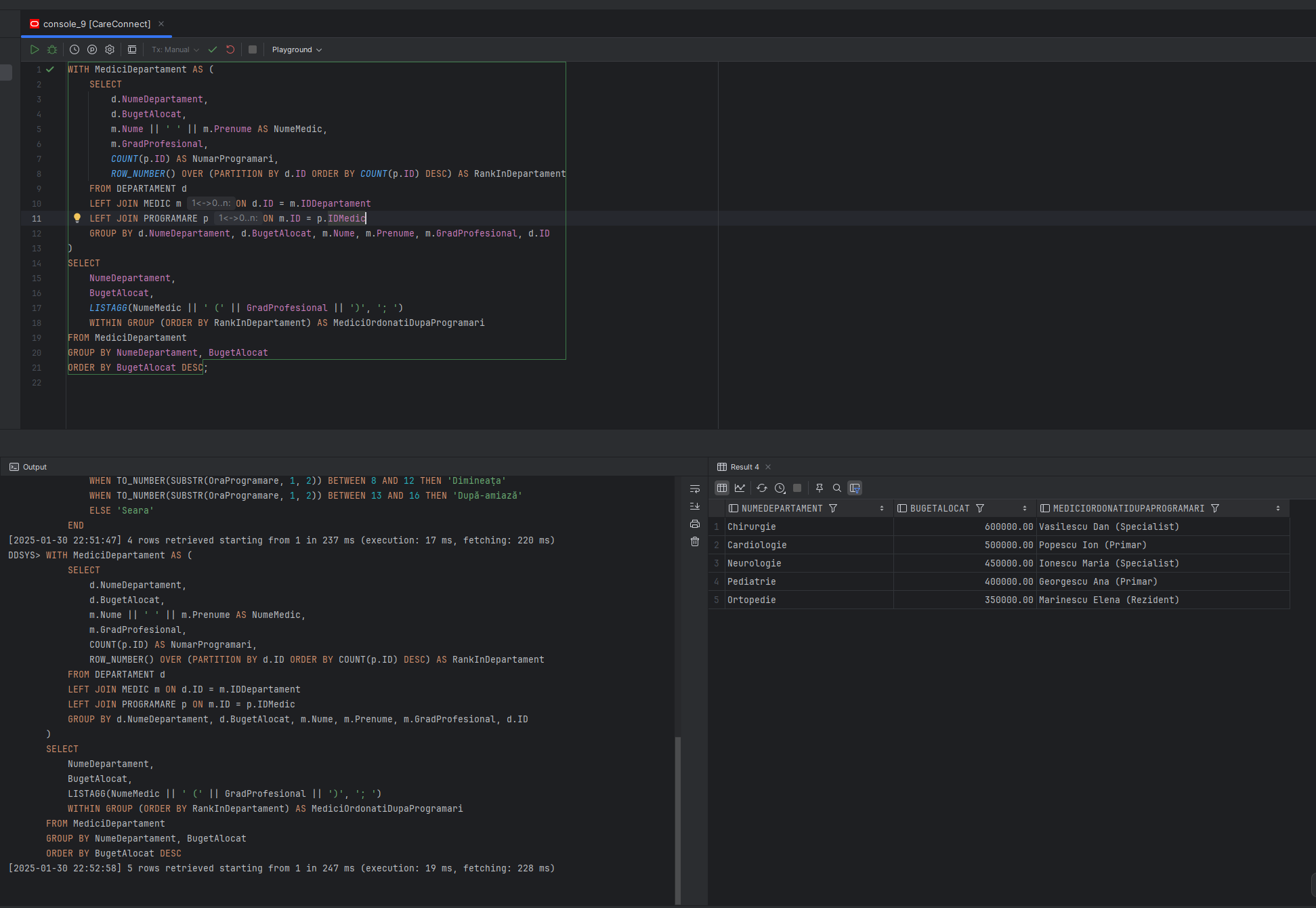
LISTAGG(NumeMedic || ' (' || GradProfesional || ')', '; ')

WITHIN GROUP (ORDER BY RankInDepartament) AS MediciOrdonatiDupaProgramari

FROM MediciDepartament

GROUP BY NumeDepartament, BugetAlocat

ORDER BY BugetAlocat DESC;



1. **Interogare care identifică medicii care au toate specializările relevante pentru departamentul lor (folosind operatorul DIVISION prin NOT EXISTS). Utilizează JOIN-uri pentru conectarea tabelelor MEDIC, DEPARTAMENT și SPECIALIZARE, și subinterogări pentru a verifica dacă medicul are toate specializările necesare departamentului său. Demonstrează utilizarea operatorului DIVISION, JOIN-uri multiple și subinterogări în WHERE.**

WITH RetetaCosturi AS (

SELECT

d.NumeDepartament,

r.ID,

SUM(rm.Cantitate \* m.PretUnitar) AS CostTotal,

COUNT(DISTINCT m.ID) AS NumarMedicamente

FROM DEPARTAMENT d

JOIN MEDIC med ON d.ID = med.IDDepartament

JOIN PROGRAMARE p ON med.ID = p.IDMedic

JOIN CONSULTATIE c ON p.ID = c.IDProgramare

JOIN RETETA r ON c.ID = r.IDConsultatie

JOIN RETETA\_MEDICAMENT rm ON r.ID = rm.IDReteta

JOIN MEDICAMENT m ON rm.IDMedicament = m.ID

GROUP BY d.NumeDepartament, r.ID

)

SELECT

NumeDepartament,

COUNT(ID) AS NumarRetete,

ROUND(AVG(CostTotal), 2) AS CostMediuReteta,

MAX(CostTotal) AS CostMaximReteta,

ROUND(AVG(NumarMedicamente), 1) AS MedieMedicamentePerReteta,

SUM(CostTotal) AS CostTotalDepartament

FROM RetetaCosturi

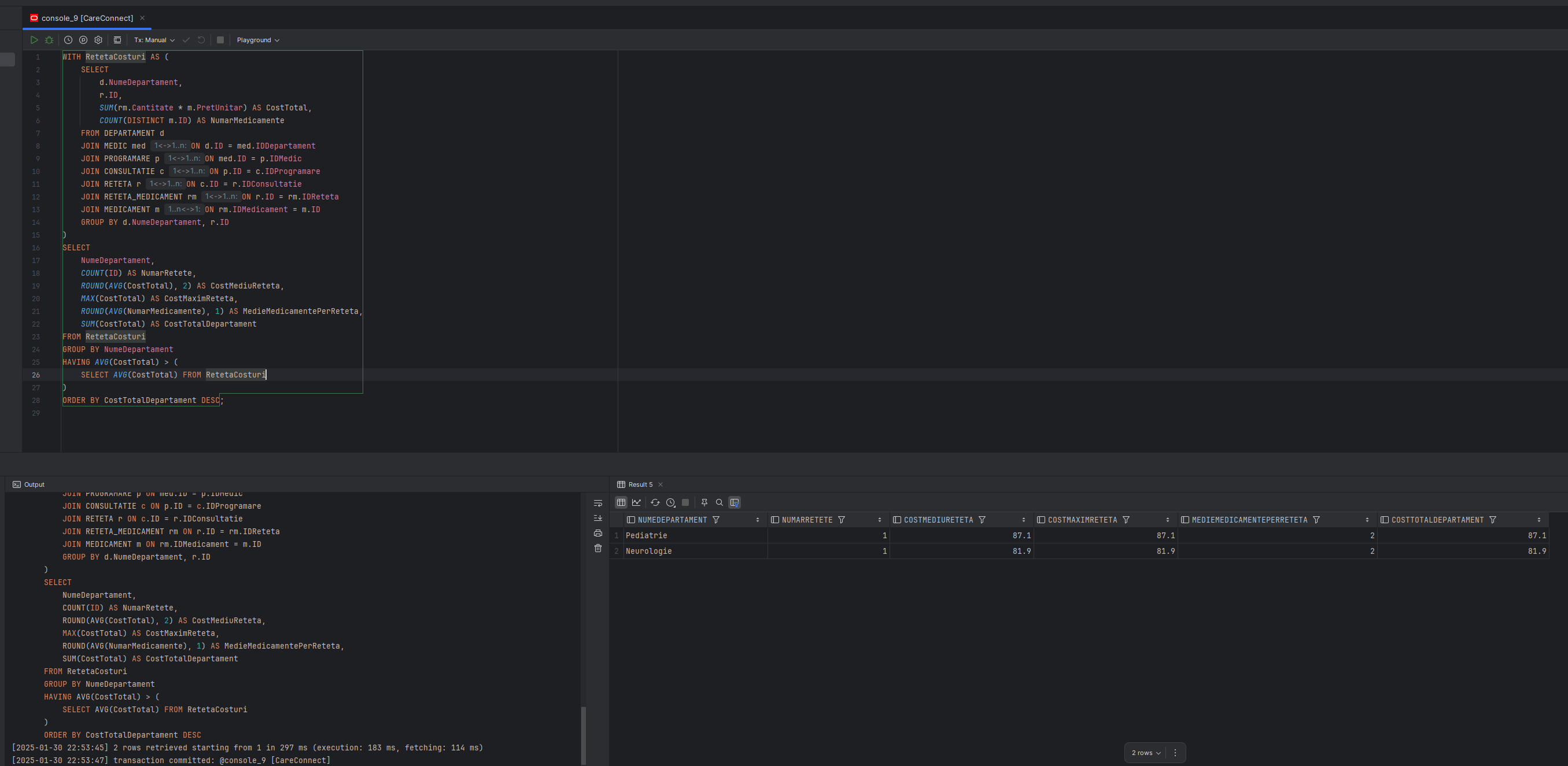
GROUP BY NumeDepartament

HAVING AVG(CostTotal) > (

SELECT AVG(CostTotal) FROM RetetaCosturi

)

ORDER BY CostTotalDepartament DESC;



1. **Interogare care analizează medicamentele prescrise, calculând statistici despre frecvența prescrierii, costurile asociate și stocuri. Utilizează GROUP BY și HAVING pentru agregare, DECODE și CASE pentru categorizare, JOIN-uri multiple pentru conectarea informațiilor despre rețete și medicamente, funcții de agregare (COUNT, SUM, AVG) și DENSE\_RANK pentru clasificarea popularității medicamentelor.**

SELECT

p.Nume || ' ' || p.Prenume AS NumePacient,

TO\_CHAR(p.DataNasterii, 'YYYY-MM-DD') AS DataNastere,

NVL(LISTAGG(DISTINCT a.Denumire, ', ') WITHIN GROUP (ORDER BY a.Denumire), 'Fără alergii') AS ListaAlergii,

COUNT(DISTINCT c.ID) AS NumarConsultatii,

MAX(c.DataConsultatie) AS UltimaConsultatie,

CASE

WHEN COUNT(DISTINCT c.ID) = 0 THEN 'Fără consultații'

WHEN MAX(c.DataConsultatie) < ADD\_MONTHS(SYSDATE, -6) THEN 'Inactiv'

ELSE 'Activ'

END AS StatusPacient

FROM PACIENT p

LEFT JOIN PACIENT\_ALERGIE pa ON p.ID = pa.IDPacient

LEFT JOIN ALERGIE a ON pa.IDAlergie = a.ID

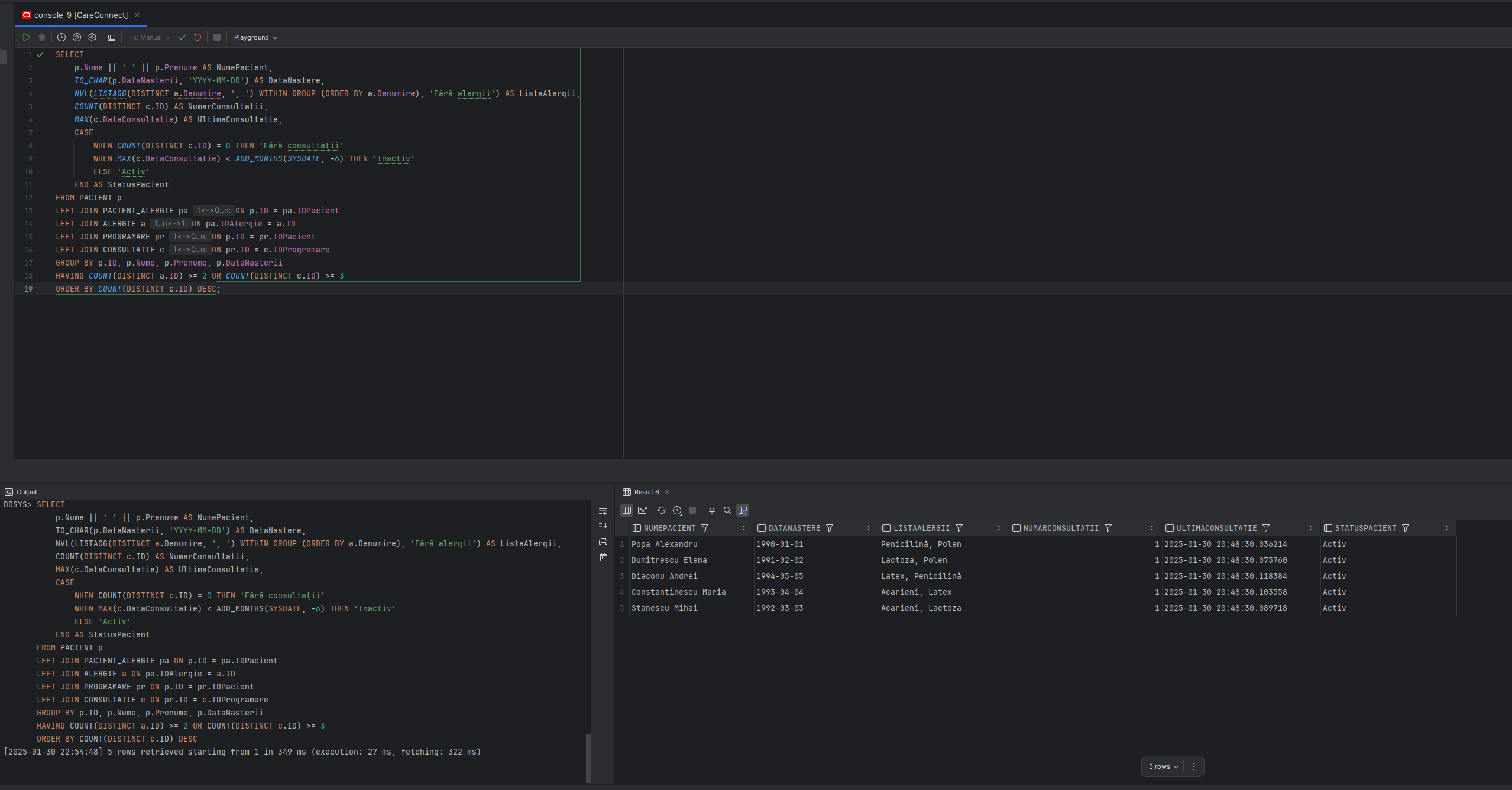
LEFT JOIN PROGRAMARE pr ON p.ID = pr.IDPacient

LEFT JOIN CONSULTATIE c ON pr.ID = c.IDProgramare

GROUP BY p.ID, p.Nume, p.Prenume, p.DataNasterii

HAVING COUNT(DISTINCT a.ID) >= 2 OR COUNT(DISTINCT c.ID) >= 3

ORDER BY COUNT(DISTINCT c.ID) DESC;



1. **Interogare care identifică medicii și specializările lor, împreună cu numărul de consultații și programări. Utilizează JOIN-uri multiple pentru a conecta tabelele MEDIC, SPECIALIZARE și DEPARTAMENT, funcții de agregare (COUNT), GROUP BY pentru gruparea rezultatelor, și ORDER BY pentru sortare. Include și funcții pentru manipularea șirurilor de caractere (concatenare) și CASE pentru categorizarea medicilor după experiență.**

SELECT

m.Denumire AS NumeMedicament,

m.FormaFarmaceutica,

COUNT(DISTINCT r.ID) AS NumarRetete,

SUM(rm.Cantitate) AS CantitateaTotala,

ROUND(AVG(rm.DurataTratament), 1) AS DurataMedieTratament,

DECODE(m.NecesitaReteta, 1, 'Da', 'Nu') AS NecesitaReteta,

CASE

WHEN m.StocDisponibil = 0 THEN 'Stoc epuizat'

WHEN m.StocDisponibil < 10 THEN 'Stoc critic'

WHEN m.StocDisponibil < 50 THEN 'Stoc redus'

ELSE 'Stoc suficient'

END AS StatusStoc,

DENSE\_RANK() OVER (ORDER BY COUNT(DISTINCT r.ID) DESC) AS RankPopularitate

FROM MEDICAMENT m

LEFT JOIN RETETA\_MEDICAMENT rm ON m.ID = rm.IDMedicament

LEFT JOIN RETETA r ON rm.IDReteta = r.ID

WHERE UPPER(m.SubstantaActiva) NOT LIKE '%PENICILINA%'

GROUP BY

m.Denumire,

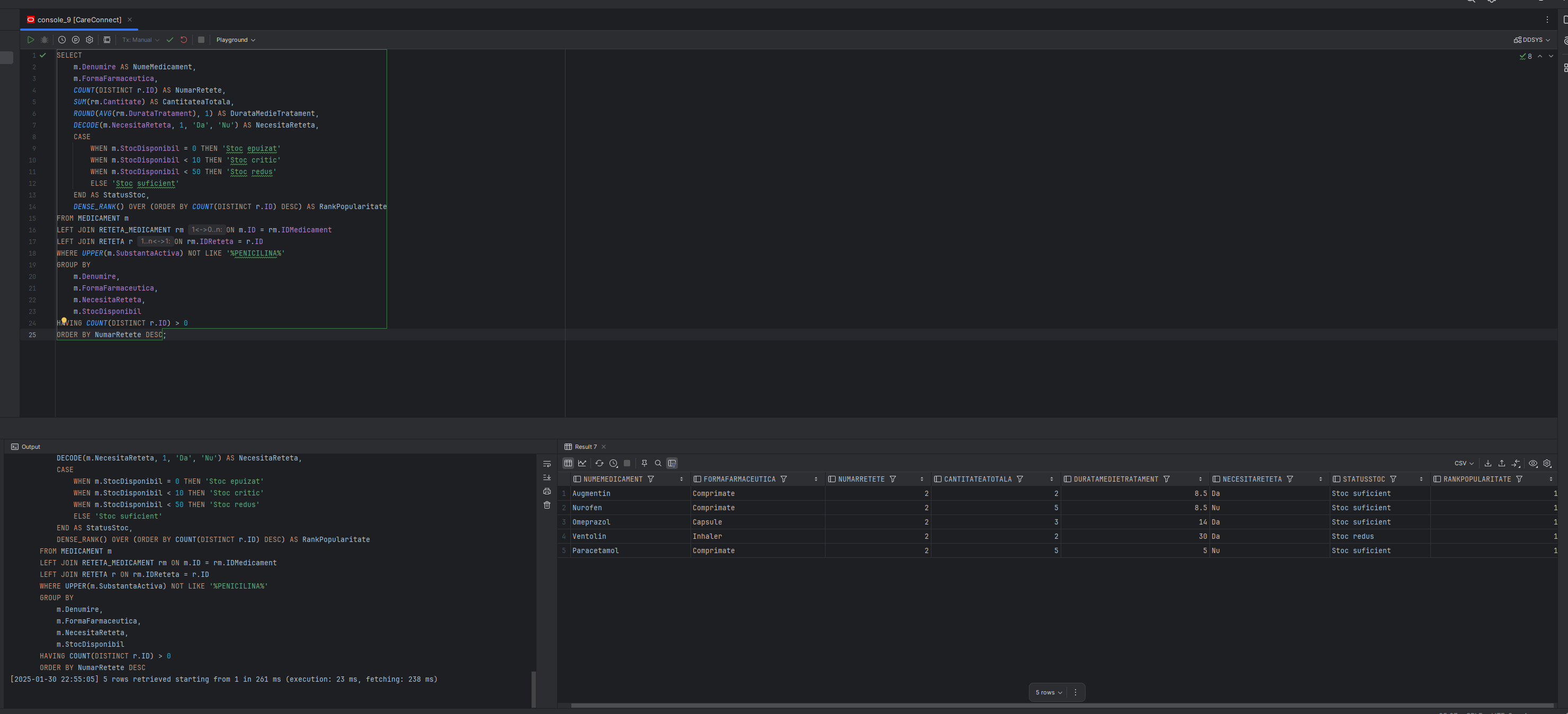
m.FormaFarmaceutica,

m.NecesitaReteta,

m.StocDisponibil

HAVING COUNT(DISTINCT r.ID) > 0

ORDER BY NumarRetete DESC;



1. **Interogare care identifică medicii care au toate specializările relevante pentru departamentul lor, folosind operatorul DIVISION (implementat prin NOT EXISTS). Utilizează JOIN-uri multiple pentru a conecta tabelele MEDIC, DEPARTAMENT și SPECIALIZARE, și folosește LIKE pentru a verifica corespondența între numele departamentului și specializări. Exemplifică utilizarea subinterogărilor în WHERE și operații pe mulțimi.**

WITH IntervaleOrare AS (

SELECT

m.ID AS IDMedic,

m.Nume || ' ' || m.Prenume AS NumeMedic,

TO\_CHAR(p.DataProgramare, 'DY') AS Zi,

CASE

WHEN SUBSTR(p.OraProgramare, 1, 2) BETWEEN '08' AND '12' THEN 'Dimineața'

WHEN SUBSTR(p.OraProgramare, 1, 2) BETWEEN '13' AND '16' THEN 'După-amiază'

ELSE 'Seara'

END AS IntervalOrar,

COUNT(\*) AS NumarProgramari

FROM MEDIC m

JOIN PROGRAMARE p ON m.ID = p.IDMedic

WHERE p.DataProgramare BETWEEN TRUNC(SYSDATE) AND TRUNC(SYSDATE) + 30

GROUP BY

m.ID,

m.Nume,

m.Prenume,

TO\_CHAR(p.DataProgramare, 'DY'),

CASE

WHEN SUBSTR(p.OraProgramare, 1, 2) BETWEEN '08' AND '12' THEN 'Dimineața'

WHEN SUBSTR(p.OraProgramare, 1, 2) BETWEEN '13' AND '16' THEN 'După-amiază'

ELSE 'Seara'

END

)

SELECT

NumeMedic,

Zi,

MAX(DECODE(IntervalOrar, 'Dimineața', NumarProgramari, 0)) AS Dimineata,

MAX(DECODE(IntervalOrar, 'După-amiază', NumarProgramari, 0)) AS DupaAmiaza,

MAX(DECODE(IntervalOrar, 'Seara', NumarProgramari, 0)) AS Seara,

SUM(NumarProgramari) AS TotalPeZi

FROM IntervaleOrare

GROUP BY NumeMedic, Zi

ORDER BY

CASE Zi

WHEN 'MON' THEN 1

WHEN 'TUE' THEN 2

WHEN 'WED' THEN 3

WHEN 'THU' THEN 4

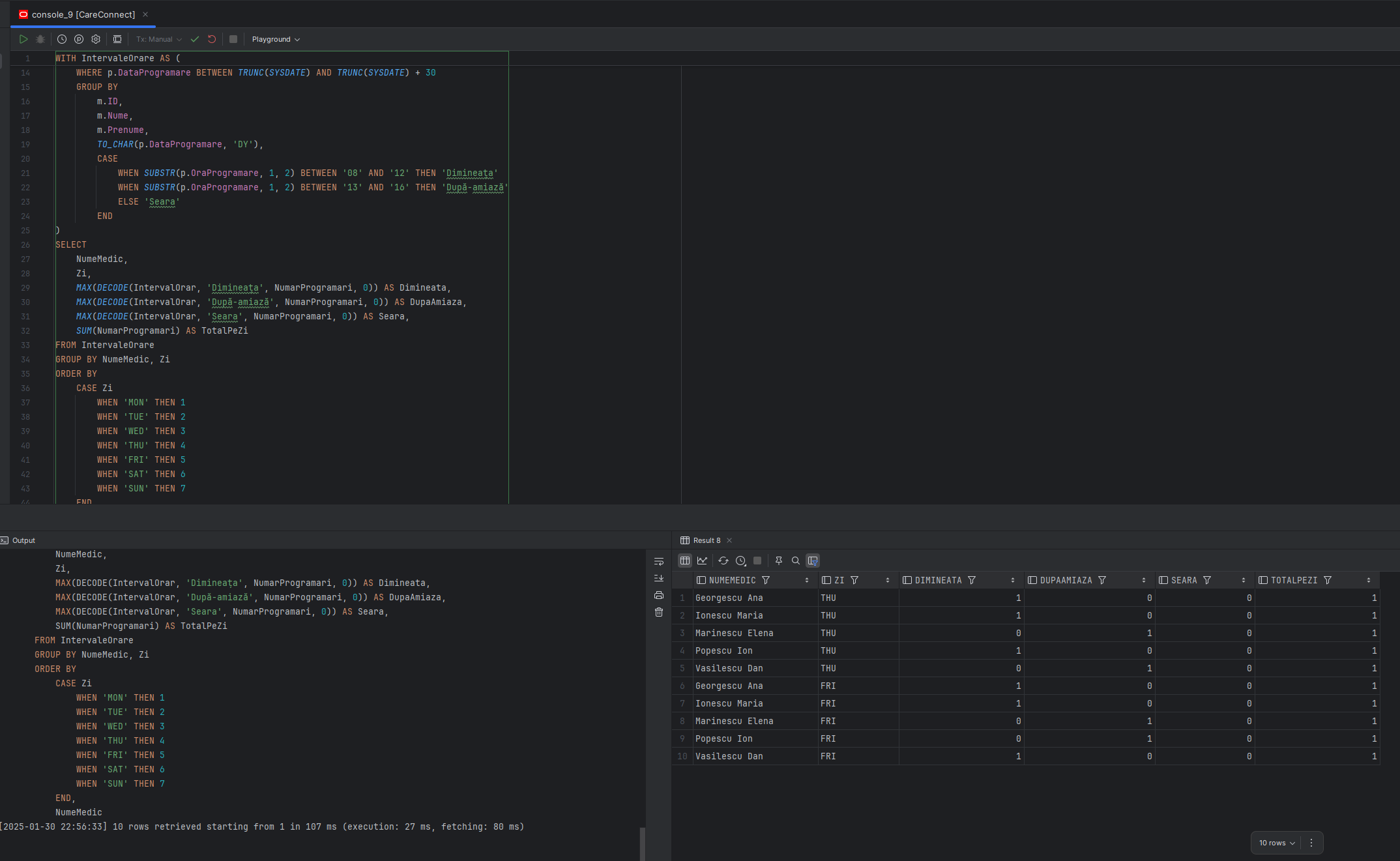
WHEN 'FRI' THEN 5

WHEN 'SAT' THEN 6

WHEN 'SUN' THEN 7

END,

NumeMedic;



1. **Interogare care identifică medicii și specializările lor, folosind LISTAGG pentru concatenarea specializărilor și LEFT JOIN pentru a include toți medicii. Utilizează funcții de agregare (COUNT), GROUP BY pentru grupare pe medici, și ORDER BY pentru ordonarea rezultatelor după numărul de specializări. Oferă o vedere de ansamblu asupra pregătirii profesionale a medicilor din fiecare departament.**

WITH ConsultatiiDepartamente AS (

SELECT

d.NumeDepartament,

d.BugetAlocat,

COUNT(DISTINCT m.ID) AS NumarMedici,

COUNT(DISTINCT c.ID) AS NumarConsultatii,

COUNT(DISTINCT r.ID) AS NumarRetete,

ROUND(AVG(c.Urgenta), 2) AS UrgentaMedie,

COUNT(DISTINCT CASE WHEN c.Urgenta >= 4 THEN c.ID END) AS ConsultatiiUrgente

FROM DEPARTAMENT d

LEFT JOIN MEDIC m ON d.ID = m.IDDepartament

LEFT JOIN PROGRAMARE p ON m.ID = p.IDMedic

LEFT JOIN CONSULTATIE c ON p.ID = c.IDProgramare

LEFT JOIN RETETA r ON c.ID = r.IDConsultatie

WHERE c.DataConsultatie >= ADD\_MONTHS(SYSDATE, -3)

GROUP BY d.NumeDepartament, d.BugetAlocat

)

SELECT

cd.NumeDepartament,

cd.NumarMedici,

cd.NumarConsultatii,

cd.ConsultatiiUrgente,

ROUND(cd.ConsultatiiUrgente \* 100.0 / NULLIF(cd.NumarConsultatii, 0), 1) AS ProcentUrgente,

cd.UrgentaMedie,

cd.NumarRetete,

ROUND(cd.BugetAlocat / NULLIF(cd.NumarConsultatii, 0), 2) AS CostMediuPerConsultatie,

CASE

WHEN cd.ConsultatiiUrgente > 10 THEN 'DEPARTAMENT CRITIC'

WHEN cd.ConsultatiiUrgente > 5 THEN 'DEPARTAMENT SOLICITAT'

ELSE 'DEPARTAMENT STABIL'

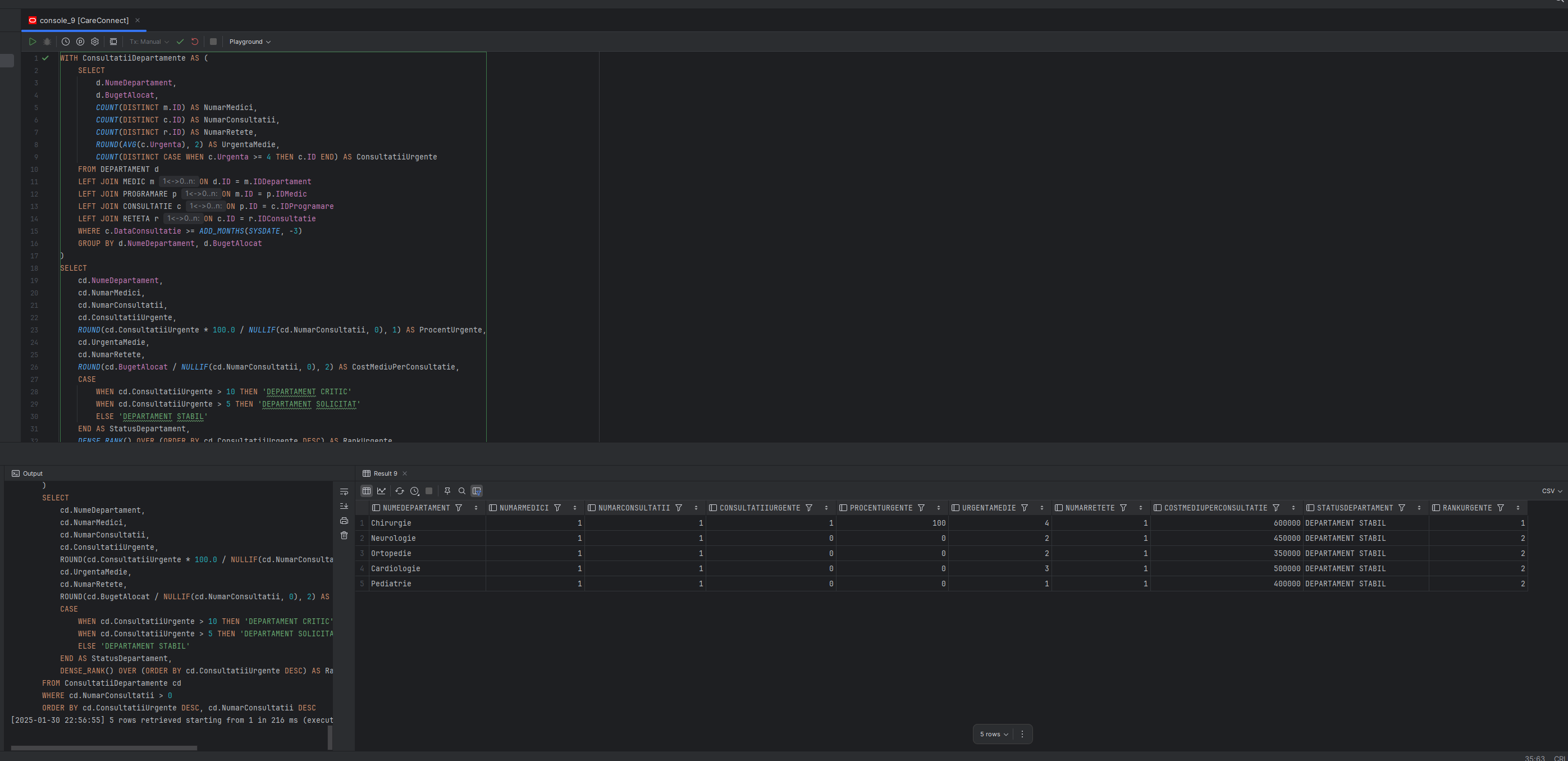
END AS StatusDepartament,

DENSE\_RANK() OVER (ORDER BY cd.ConsultatiiUrgente DESC) AS RankUrgente

FROM ConsultatiiDepartamente cd

WHERE cd.NumarConsultatii > 0

ORDER BY cd.ConsultatiiUrgente DESC, cd.NumarConsultatii DESC;



1. **Interogare care identifică medicii și specializările lor, folosind LISTAGG pentru concatenarea specializărilor și LEFT JOIN pentru a include toți medicii. Utilizează funcții de agregare (COUNT), GROUP BY pentru gruparea pe medici, și ORDER BY pentru ordonarea rezultatelor după numărul de specializări. Oferă o vedere de ansamblu asupra pregătirii profesionale a medicilor din sistem.**

WITH StatisticiMedic AS (

SELECT

m.ID,

m.Nume || ' ' || m.Prenume AS NumeMedic,

d.NumeDepartament,

COUNT(DISTINCT p.ID) AS TotalProgramari,

COUNT(DISTINCT c.ID) AS TotalConsultatii,

COUNT(DISTINCT r.ID) AS TotalRetete,

ROUND(AVG(c.Urgenta), 2) AS MedieUrgenta,

COUNT(DISTINCT ms.IDSpecializare) AS NumarSpecializari

FROM MEDIC m

JOIN DEPARTAMENT d ON m.IDDepartament = d.ID

LEFT JOIN PROGRAMARE p ON m.ID = p.IDMedic

LEFT JOIN CONSULTATIE c ON p.ID = c.IDProgramare

LEFT JOIN RETETA r ON c.ID = r.IDConsultatie

LEFT JOIN MEDIC\_SPECIALIZARE ms ON m.ID = ms.IDMedic

WHERE p.DataProgramare >= ADD\_MONTHS(SYSDATE, -6)

GROUP BY m.ID, m.Nume, m.Prenume, d.NumeDepartament

)

SELECT

NumeMedic,

NumeDepartament,

TotalProgramari,

TotalConsultatii,

ROUND(TotalConsultatii \* 100.0 / NULLIF(TotalProgramari, 0), 1) AS RataFinalizare,

TotalRetete,

MedieUrgenta,

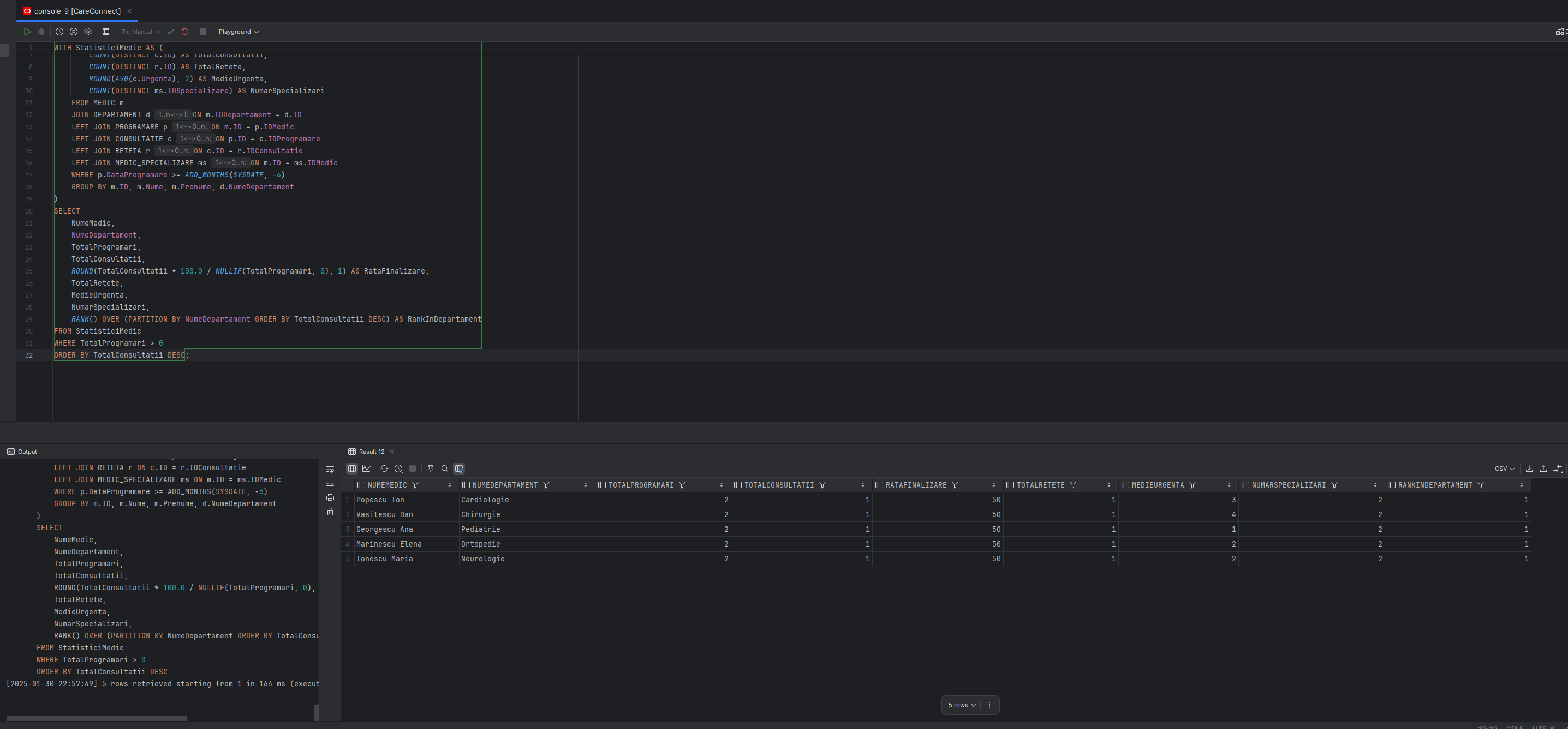
NumarSpecializari,

RANK() OVER (PARTITION BY NumeDepartament ORDER BY TotalConsultatii DESC) AS RankInDepartament

FROM StatisticiMedic

WHERE TotalProgramari > 0

ORDER BY TotalConsultatii DESC;



1. **Interogare care analizează distribuția programărilor pe zile ale săptămânii și intervale orare (Dimineața, După-amiază, Seara). Folosește funcții de dată (TO\_CHAR), CASE pentru categorizarea intervalelor orare, funcții de agregare (COUNT), calcule procentuale și GROUP BY pentru a genera statistici despre frecvența programărilor. Rezultatele sunt ordonate după ziua săptămânii pentru o vizualizare clară a pattern-urilor de programare.**

WITH PrescrieriMedicament AS (

SELECT

m.ID,

m.Denumire,

m.SubstantaActiva,

m.FormaFarmaceutica,

COUNT(DISTINCT rm.IDReteta) AS NumarPrescrieri,

SUM(rm.Cantitate) AS CantitatePrescrisa,

AVG(rm.DurataTratament) AS DurataMedieTratament,

COUNT(DISTINCT r.IDConsultatie) AS NumarConsultatii,

COUNT(DISTINCT p.IDPacient) AS NumarPacienti

FROM MEDICAMENT m

LEFT JOIN RETETA\_MEDICAMENT rm ON m.ID = rm.IDMedicament

LEFT JOIN RETETA r ON rm.IDReteta = r.ID

LEFT JOIN CONSULTATIE c ON r.IDConsultatie = c.ID

LEFT JOIN PROGRAMARE p ON c.IDProgramare = p.ID

GROUP BY m.ID, m.Denumire, m.SubstantaActiva, m.FormaFarmaceutica

)

SELECT

Denumire,

SubstantaActiva,

FormaFarmaceutica,

NumarPrescrieri,

CantitatePrescrisa,

ROUND(DurataMedieTratament, 1) AS DurataMedieTratament,

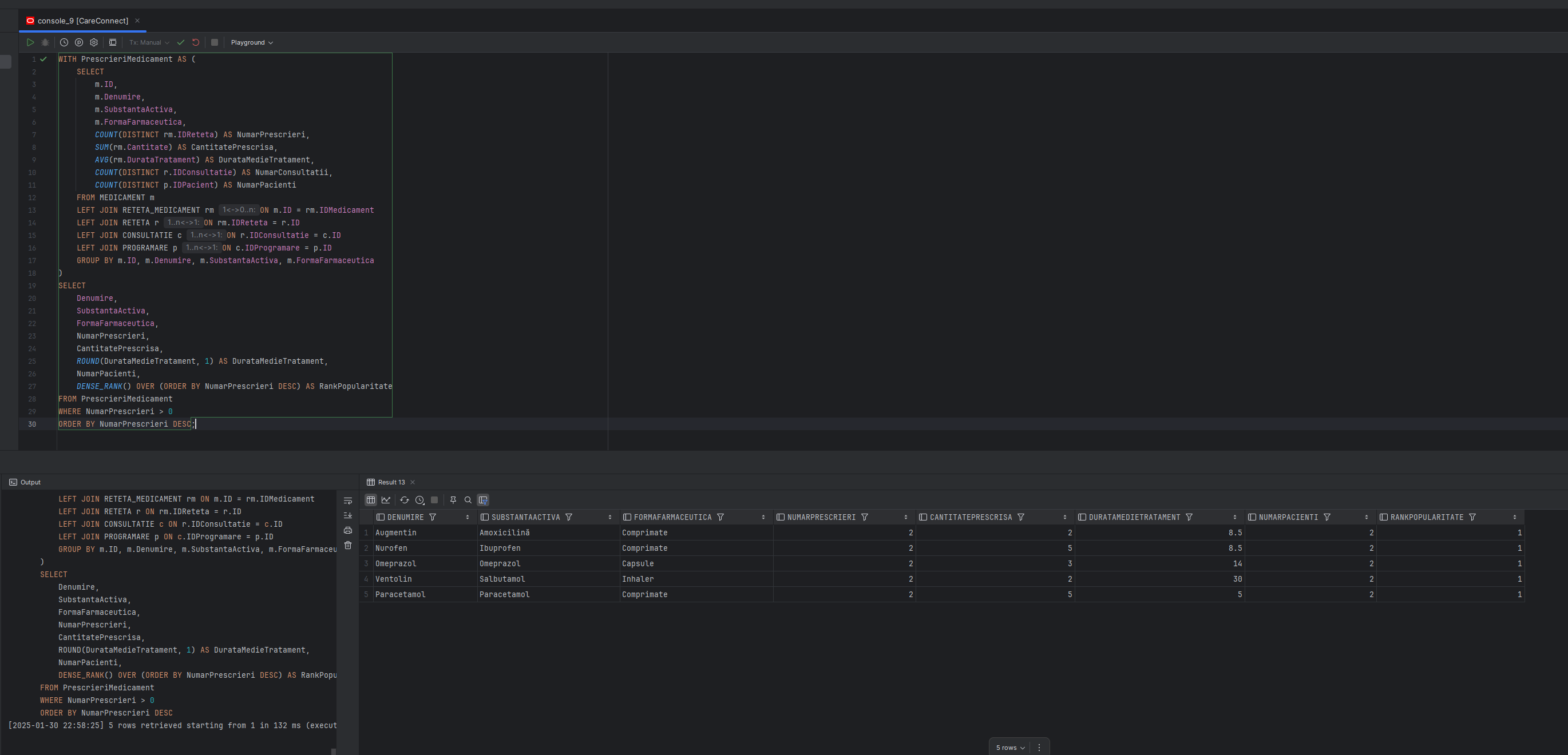
NumarPacienti,

DENSE\_RANK() OVER (ORDER BY NumarPrescrieri DESC) AS RankPopularitate

FROM PrescrieriMedicament

WHERE NumarPrescrieri > 0

ORDER BY NumarPrescrieri DESC;



1. **Interogare care analizează programările pe zile și ore, calculând rata de prezență a pacienților și urgența medie. Folosește funcții de dată (TO\_CHAR), GROUP BY cu HAVING, funcții de agregare (COUNT, AVG), JOIN-uri multiple și CASE pentru ordonarea zilelor săptămânii. Oferă o perspectivă asupra momentelor cu cel mai mare grad de ocupare și a eficienței programărilor.**

SELECT

TO\_CHAR(p.DataProgramare, 'DY') AS ZiSaptamana,

TO\_CHAR(p.DataProgramare, 'HH24:MI') AS Ora,

COUNT(\*) AS TotalProgramari,

COUNT(c.ID) AS ProgramariOnorate,

ROUND(COUNT(c.ID) \* 100.0 / COUNT(\*), 1) AS RataPrezenta,

AVG(c.Urgenta) AS UrgentaMedie,

COUNT(DISTINCT m.ID) AS NumarMedici,

COUNT(DISTINCT d.ID) AS NumarDepartamente

FROM PROGRAMARE p

LEFT JOIN CONSULTATIE c ON p.ID = c.IDProgramare

JOIN MEDIC m ON p.IDMedic = m.ID

JOIN DEPARTAMENT d ON m.IDDepartament = d.ID

WHERE p.DataProgramare BETWEEN ADD\_MONTHS(SYSDATE, -1) AND SYSDATE

GROUP BY TO\_CHAR(p.DataProgramare, 'DY'), TO\_CHAR(p.DataProgramare, 'HH24:MI')

HAVING COUNT(\*) >= 5

ORDER BY

CASE TO\_CHAR(p.DataProgramare, 'DY')

WHEN 'MON' THEN 1 WHEN 'TUE' THEN 2

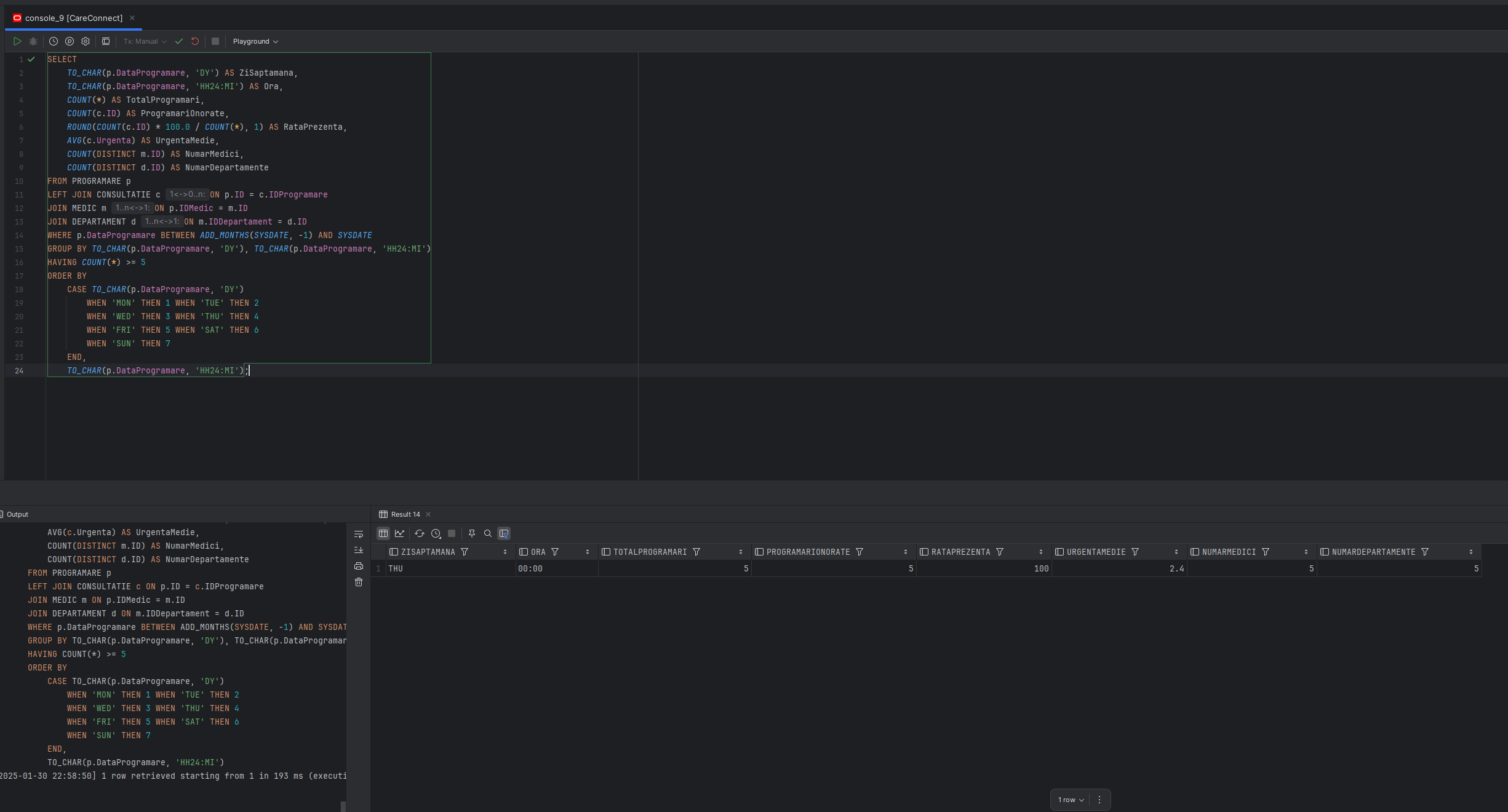
WHEN 'WED' THEN 3 WHEN 'THU' THEN 4

WHEN 'FRI' THEN 5 WHEN 'SAT' THEN 6

WHEN 'SUN' THEN 7

END,

TO\_CHAR(p.DataProgramare, 'HH24:MI');



1. **Interogare care calculează statistici despre eficiența fiecărui departament, incluzând numărul de medici, programări, consultații și bugetul alocat. Folosește CTE pentru structurare, JOIN-uri (LEFT JOIN), funcții de agregare (COUNT, SUM, AVG), CASE pentru categorizare, NULLIF pentru evitarea împărțirii la zero, și RANK pentru clasificarea departamentelor după activitate. Rezultatele sunt ordonate după rata de finalizare a consultațiilor.**

WITH EficientaDepartament AS (

SELECT

d.ID,

d.NumeDepartament,

d.BugetAlocat,

COUNT(DISTINCT m.ID) AS NumarMedici,

COUNT(DISTINCT p.ID) AS NumarProgramari,

COUNT(DISTINCT c.ID) AS NumarConsultatii,

COUNT(DISTINCT r.ID) AS NumarRetete,

SUM(rm.Cantitate \* med.PretUnitar) AS CostMedicamente

FROM DEPARTAMENT d

LEFT JOIN MEDIC m ON d.ID = m.IDDepartament

LEFT JOIN PROGRAMARE p ON m.ID = p.IDMedic

LEFT JOIN CONSULTATIE c ON p.ID = c.IDProgramare

LEFT JOIN RETETA r ON c.ID = r.IDConsultatie

LEFT JOIN RETETA\_MEDICAMENT rm ON r.ID = rm.IDReteta

LEFT JOIN MEDICAMENT med ON rm.IDMedicament = med.ID

GROUP BY d.ID, d.NumeDepartament, d.BugetAlocat

)

SELECT

NumeDepartament,

NumarMedici,

NumarProgramari,

NumarConsultatii,

ROUND(NumarConsultatii \* 100.0 / NULLIF(NumarProgramari, 0), 1) AS RataFinalizare,

NumarRetete,

ROUND(BugetAlocat / NULLIF(NumarConsultatii, 0), 2) AS CostPerConsultatie,

ROUND(CostMedicamente / NULLIF(NumarRetete, 0), 2) AS CostMediuReteta,

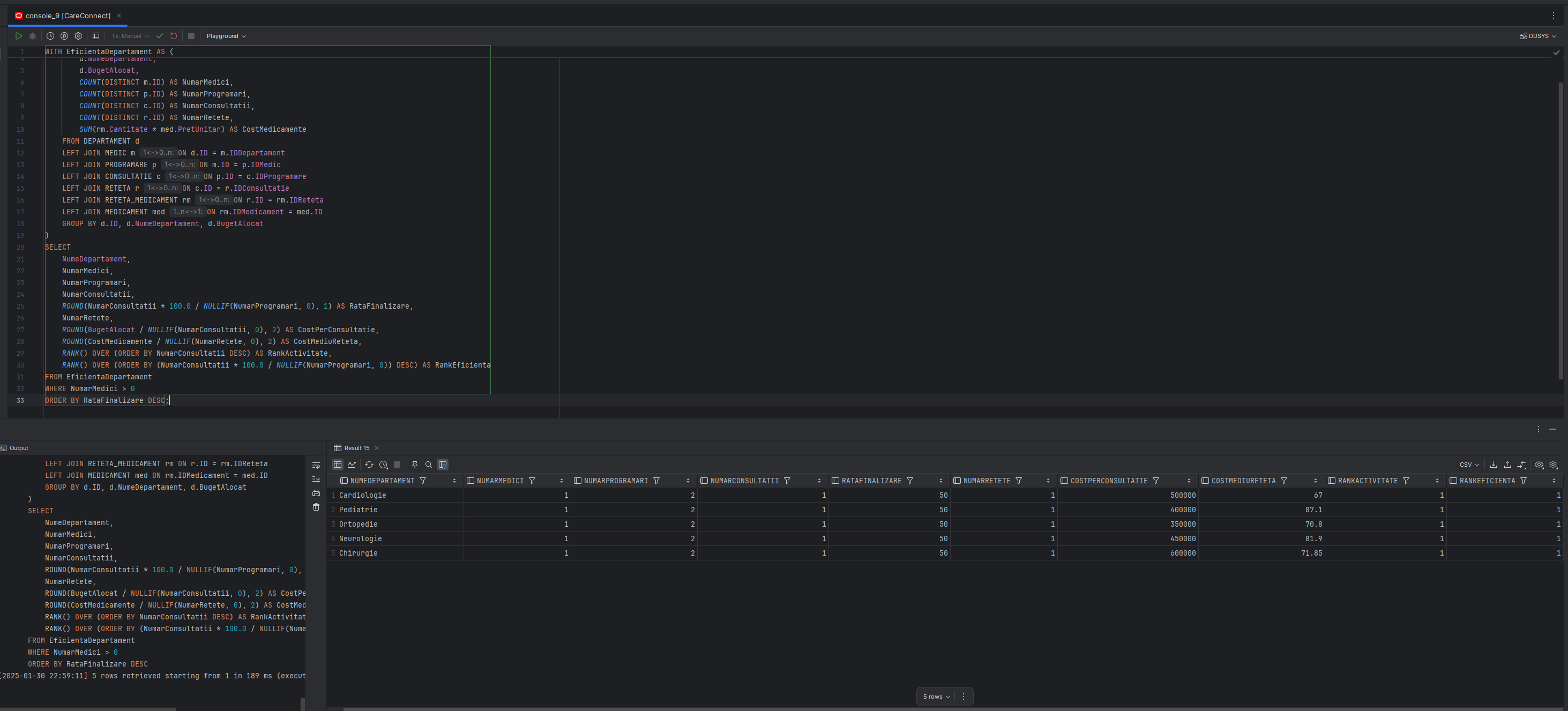
RANK() OVER (ORDER BY NumarConsultatii DESC) AS RankActivitate,

RANK() OVER (ORDER BY (NumarConsultatii \* 100.0 / NULLIF(NumarProgramari, 0)) DESC) AS RankEficienta

FROM EficientaDepartament

WHERE NumarMedici > 0

ORDER BY RataFinalizare DESC;



1. **Interogare care afișează structura ierarhică a gradelor profesionale ale medicilor (Primar -> Specialist -> Rezident), folosind START WITH și CONNECT BY. Include și numărul de medici pentru fiecare grad profesional.**

WITH IerarhieMedici AS (

SELECT DISTINCT

GradProfesional,

CASE

WHEN GradProfesional = 'Primar' THEN NULL

WHEN GradProfesional = 'Specialist' THEN 'Primar'

WHEN GradProfesional = 'Rezident' THEN 'Specialist'

END AS GradSuperior

FROM MEDIC

)

SELECT

LEVEL as Nivel,

LPAD(' ', 2 \* (LEVEL - 1)) || GradProfesional as Structura,

(

SELECT COUNT(\*)

FROM MEDIC m

WHERE m.GradProfesional = i.GradProfesional

) as NumarMedici,

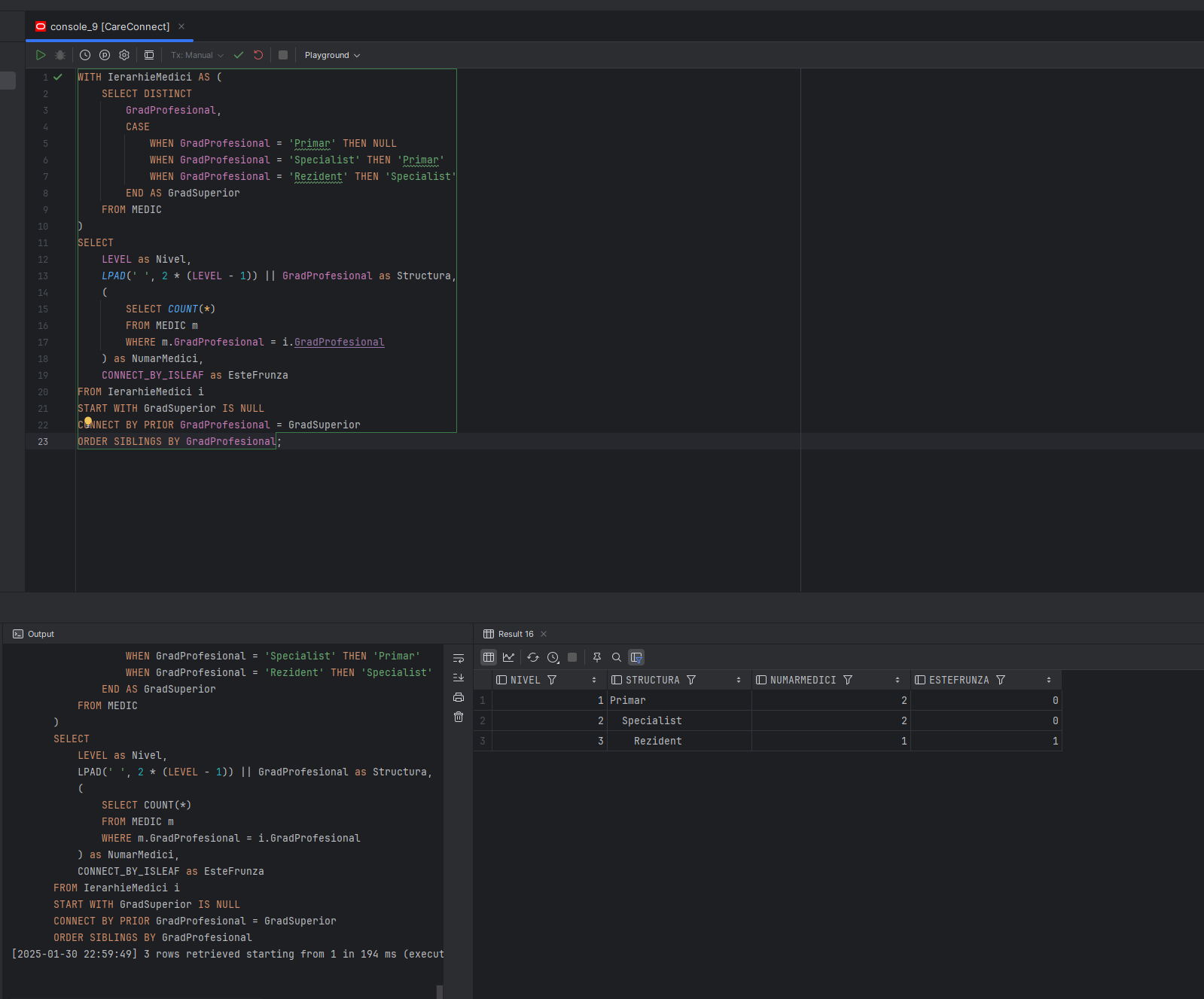
CONNECT\_BY\_ISLEAF as EsteFrunza

FROM IerarhieMedici i

START WITH GradSuperior IS NULL

CONNECT BY PRIOR GradProfesional = GradSuperior

ORDER SIBLINGS BY GradProfesional;



## 8. Creare Tabel Mesaje

CREATE TABLE MESAJE (

cod\_mesaj NUMBER PRIMARY KEY,

mesaj VARCHAR2(255),

tip\_mesaj VARCHAR2(1) CHECK (tip\_mesaj IN ('E', 'W', 'I')),

creat\_de VARCHAR2(40) NOT NULL,

creat\_la DATE NOT NULL

);

-- Creare secvență pentru cod\_mesaj

CREATE SEQUENCE seq\_mesaje START WITH 1 INCREMENT BY 1;

-- Comentarii pentru documentare

COMMENT ON TABLE MESAJE IS 'Tabel pentru stocarea mesajelor sistemului';

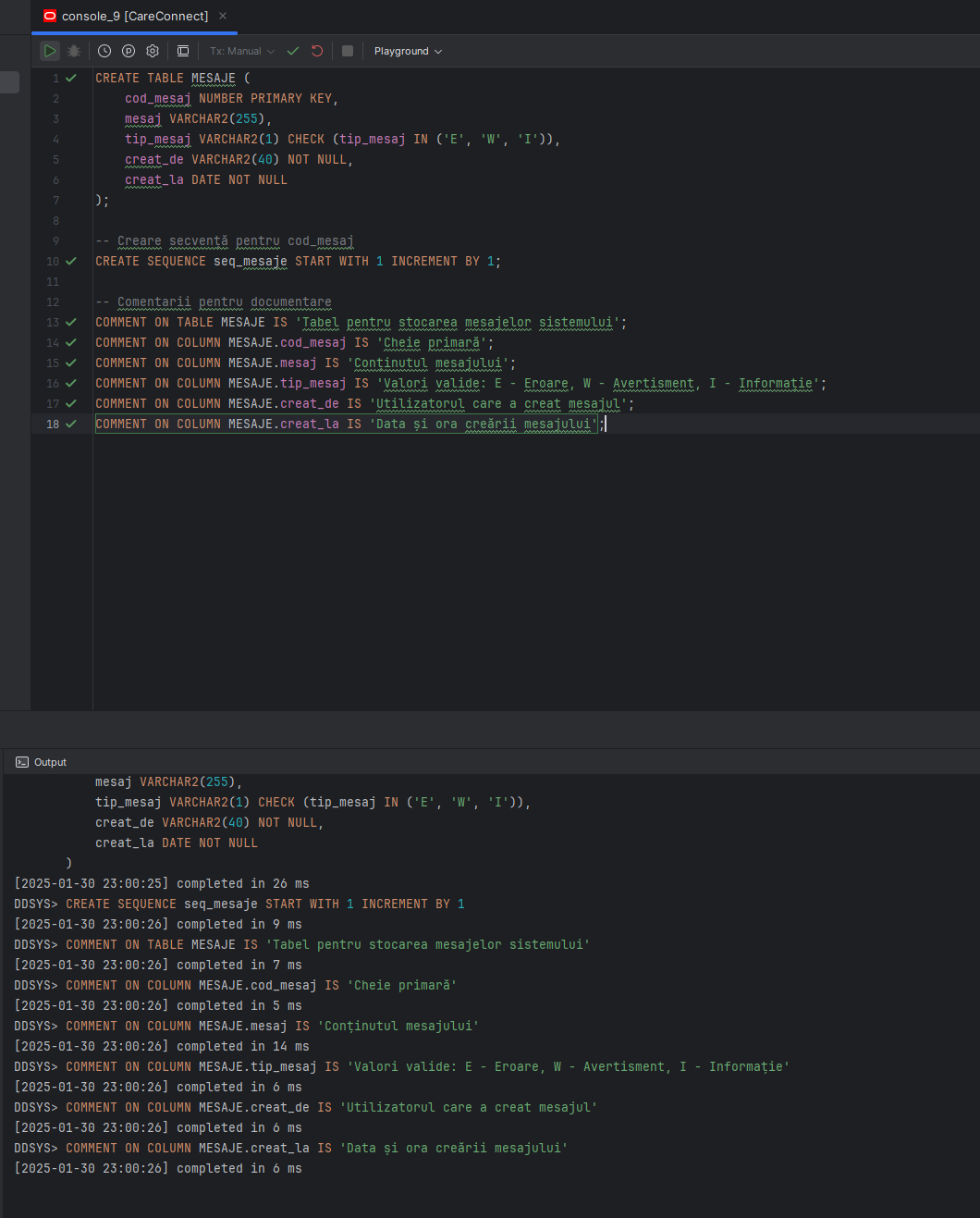
COMMENT ON COLUMN MESAJE.cod\_mesaj IS 'Cheie primară';

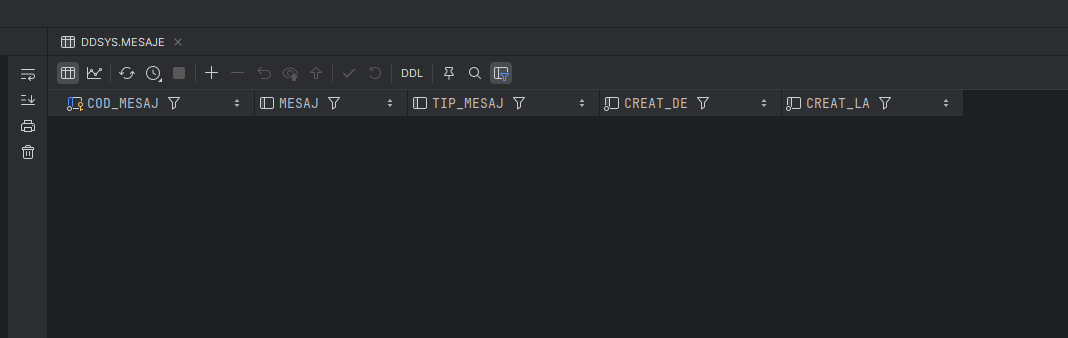
COMMENT ON COLUMN MESAJE.mesaj IS 'Conținutul mesajului';

COMMENT ON COLUMN MESAJE.tip\_mesaj IS 'Valori valide: E - Eroare, W - Avertisment, I - Informație';

COMMENT ON COLUMN MESAJE.creat\_de IS 'Utilizatorul care a creat mesajul';

COMMENT ON COLUMN MESAJE.creat\_la IS 'Data și ora creării mesajului';





## PL/SQL

Punct 1

CREATE OR REPLACE PROCEDURE gestionare\_pacienti AS

-- Tablou indexat

TYPE t\_pacienti IS TABLE OF Pacient%ROWTYPE INDEX BY PLS\_INTEGER;

v\_pacienti t\_pacienti;

-- Tablou imbricat

TYPE t\_nume IS TABLE OF VARCHAR2(100);

v\_nume t\_nume := t\_nume();

-- Vector

TYPE v\_cnp IS VARRAY(5) OF VARCHAR2(13);

v\_cnp\_uri v\_cnp := v\_cnp();

-- Pentru logging

v\_mesaj VARCHAR2(255);

BEGIN

-- Populare tablou indexat

SELECT \* BULK COLLECT INTO v\_pacienti

FROM Pacient

WHERE ROWNUM <= 5;

-- Populare tablou imbricat

FOR i IN 1..v\_pacienti.COUNT LOOP

v\_nume.EXTEND;

v\_nume(i) := v\_pacienti(i).nume || ' ' || v\_pacienti(i).prenume;

END LOOP;

-- Populare vector

v\_cnp\_uri.EXTEND(3);

FOR i IN 1..3 LOOP

v\_cnp\_uri(i) := v\_pacienti(i).cnp;

END LOOP;

-- AfiÈ™are rezultate

FOR i IN 1..v\_nume.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE('Pacient ' || i || ': ' || v\_nume(i));

END LOOP;

-- Logging succes

v\_mesaj := 'Procedura gestionare\_pacienti executatÄƒ cu succes';

INSERT INTO MESAJE (cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)

VALUES (seq\_mesaje.NEXTVAL, v\_mesaj, 'I', USER, SYSDATE);

EXCEPTION

WHEN OTHERS THEN

-- Logging eroare

v\_mesaj := 'Eroare Ã®n procedura gestionare\_pacienti: ' || SQLERRM;

INSERT INTO MESAJE (cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)

VALUES (seq\_mesaje.NEXTVAL, v\_mesaj, 'E', USER, SYSDATE);

RAISE;

END;

/

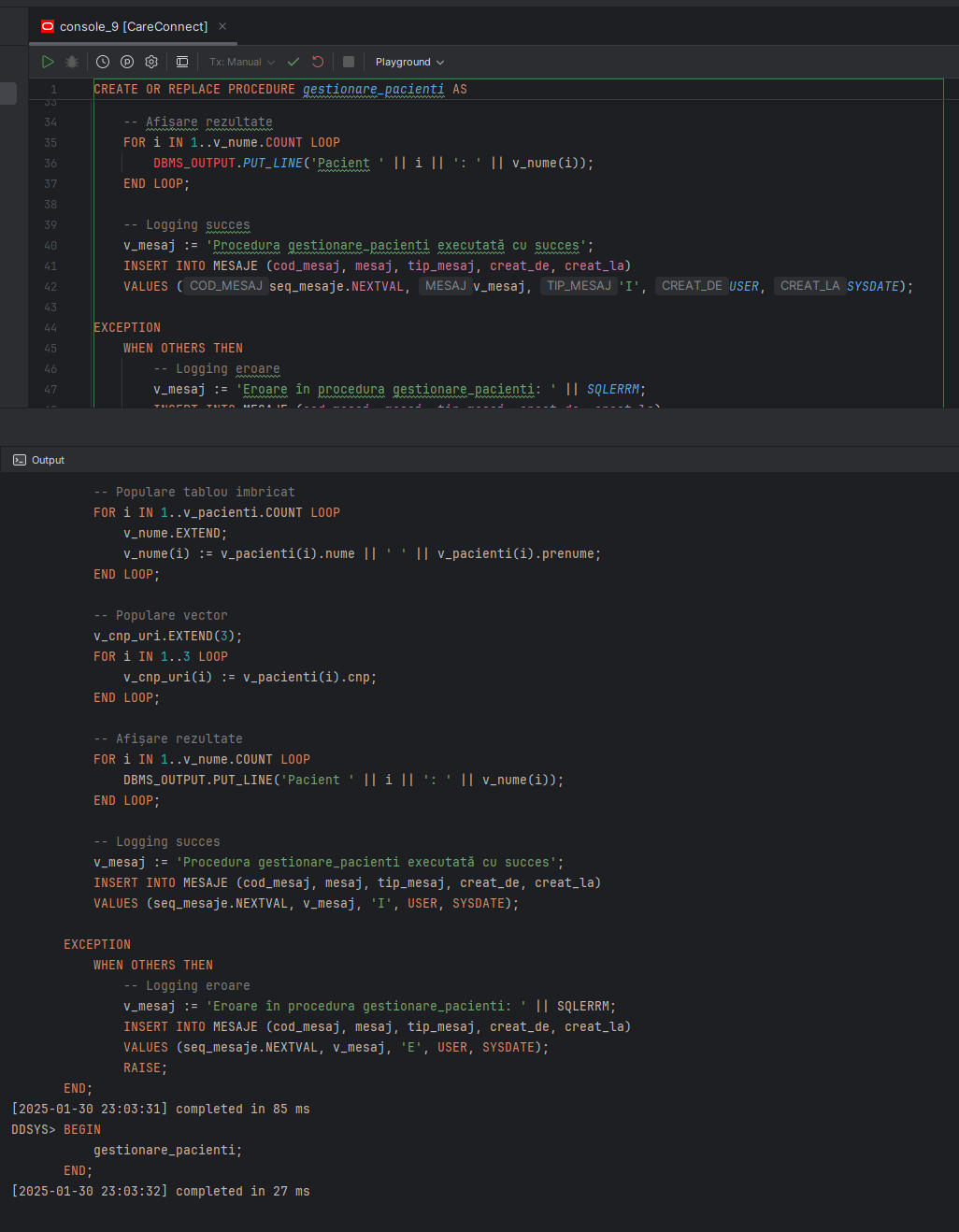
-- Apelare

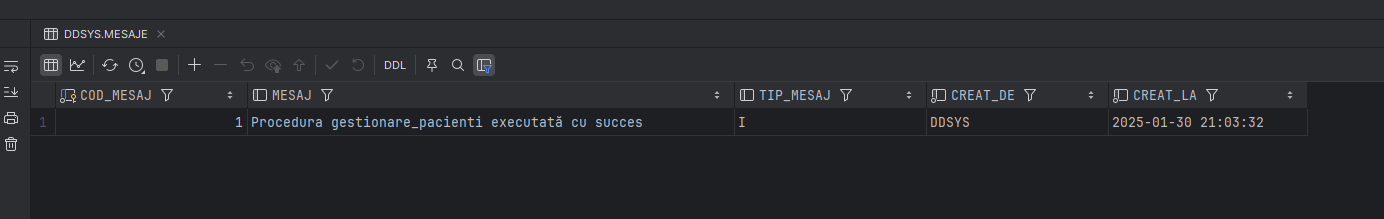
BEGIN

gestionare\_pacienti;

END;

/





Punct 2

CREATE OR REPLACE PROCEDURE vizualizare\_departamente\_medici AS

-- Primul cursor - pentru departamente

CURSOR c\_departamente IS

SELECT id, NumeDepartament

FROM Departament;

-- Al doilea cursor - parametrizat, pentru medicii din departament

CURSOR c\_medici (p\_id\_departament NUMBER) IS

SELECT Nume, Prenume, GradProfesional

FROM Medic

WHERE IDDepartament = p\_id\_departament;

v\_mesaj VARCHAR2(255);

BEGIN

-- Pentru fiecare departament

FOR dep IN c\_departamente LOOP

DBMS\_OUTPUT.PUT\_LINE('Departament: ' || dep.NumeDepartament);

-- Pentru fiecare medic din departamentul curent

FOR med IN c\_medici(dep.id) LOOP

DBMS\_OUTPUT.PUT\_LINE(' - ' || med.Nume || ' ' || med.Prenume ||

' (' || med.GradProfesional || ')');

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('-------------------');

END LOOP;

-- Logging succes

v\_mesaj := 'Procedura vizualizare\_departamente\_medici executată cu succes';

INSERT INTO MESAJE

VALUES (seq\_mesaje.NEXTVAL, v\_mesaj, 'I', USER, SYSDATE);

EXCEPTION

WHEN OTHERS THEN

-- Logging eroare

v\_mesaj := 'Eroare în procedura vizualizare\_departamente\_medici: ' || SQLERRM;

INSERT INTO MESAJE

VALUES (seq\_mesaje.NEXTVAL, v\_mesaj, 'E', USER, SYSDATE);

RAISE;

END;

/

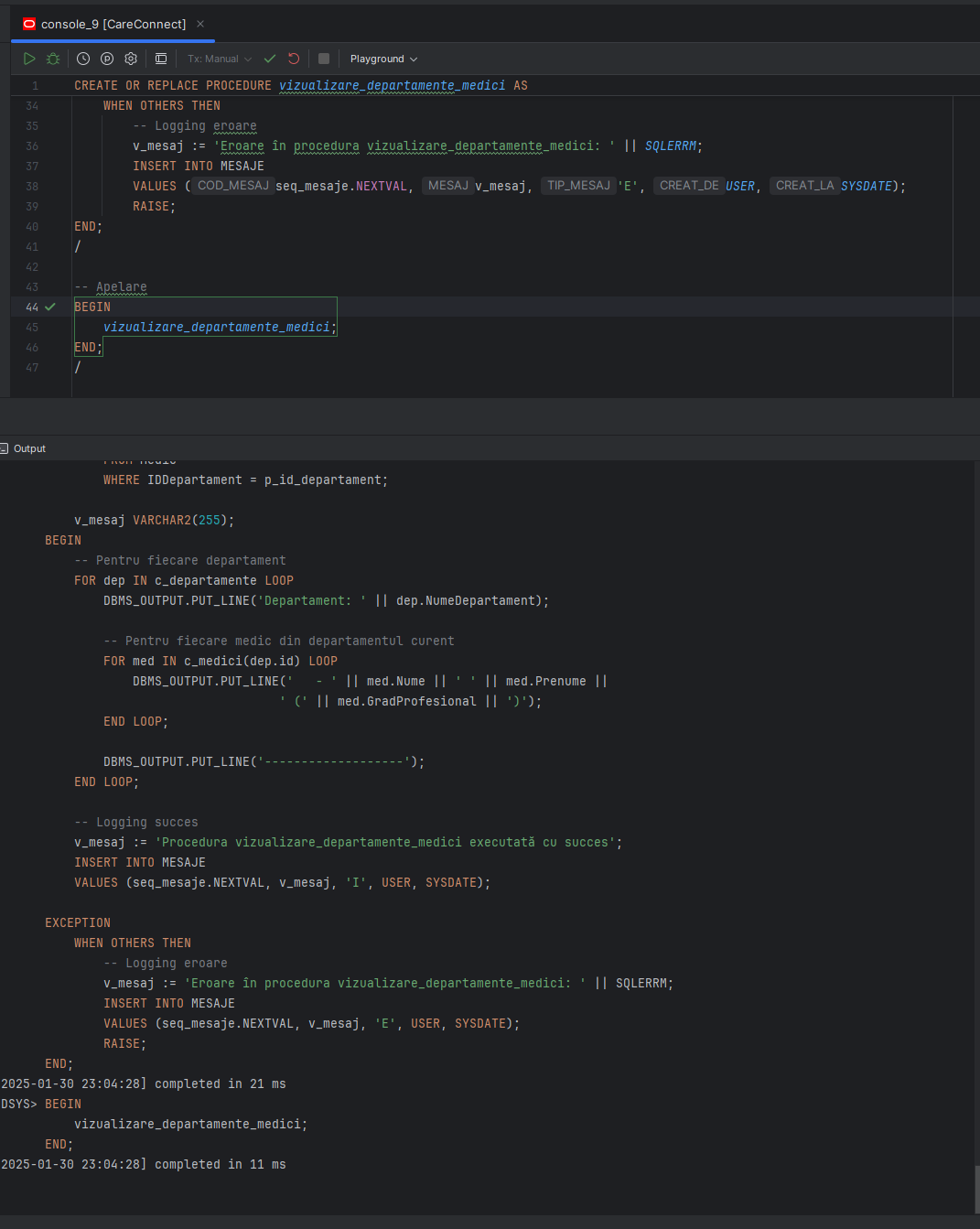
-- Apelare

BEGIN

vizualizare\_departamente\_medici;

END;

/





Punct 3

CREATE OR REPLACE FUNCTION calculeaza\_programari\_medic(

p\_id\_medic NUMBER

) RETURN NUMBER IS

v\_numar\_programari NUMBER := 0;

v\_exista\_medic NUMBER;

-- Definire excepții

e\_medic\_inexistent EXCEPTION;

e\_prea\_multe\_programari EXCEPTION;

BEGIN

-- Verifică dacă medicul există

SELECT COUNT(\*) INTO v\_exista\_medic

FROM Medic

WHERE ID = p\_id\_medic;

IF v\_exista\_medic = 0 THEN

-- Logging și raise pentru medic inexistent

INSERT INTO MESAJE (cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)

VALUES (seq\_mesaje.NEXTVAL, 'Medicul nu există', 'E', USER, SYSDATE);

RAISE e\_medic\_inexistent;

END IF;

-- Numără programările medicului

SELECT COUNT(\*) INTO v\_numar\_programari

FROM Programare

WHERE IDMedic = p\_id\_medic

AND DataProgramare >= SYSDATE - 30;

IF v\_numar\_programari > 100 THEN

-- Logging și raise pentru prea multe programări

INSERT INTO MESAJE (cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)

VALUES (seq\_mesaje.NEXTVAL, 'Prea multe programări', 'W', USER, SYSDATE);

RAISE e\_prea\_multe\_programari;

END IF;

-- Logging succes

INSERT INTO MESAJE (cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)

VALUES (seq\_mesaje.NEXTVAL, 'Calcul reușit', 'I', USER, SYSDATE);

RETURN v\_numar\_programari;

EXCEPTION

WHEN e\_medic\_inexistent THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Medicul nu există în baza de date');

WHEN e\_prea\_multe\_programari THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Prea multe programări');

WHEN OTHERS THEN

INSERT INTO MESAJE (cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)

VALUES (seq\_mesaje.NEXTVAL, 'Eroare neașteptată', 'E', USER, SYSDATE);

RAISE;

END;

/

-- Apelare pentru testare

DECLARE

v\_rezultat NUMBER;

BEGIN

v\_rezultat := calculeaza\_programari\_medic(1);

DBMS\_OUTPUT.PUT\_LINE('Număr programări: ' || v\_rezultat);

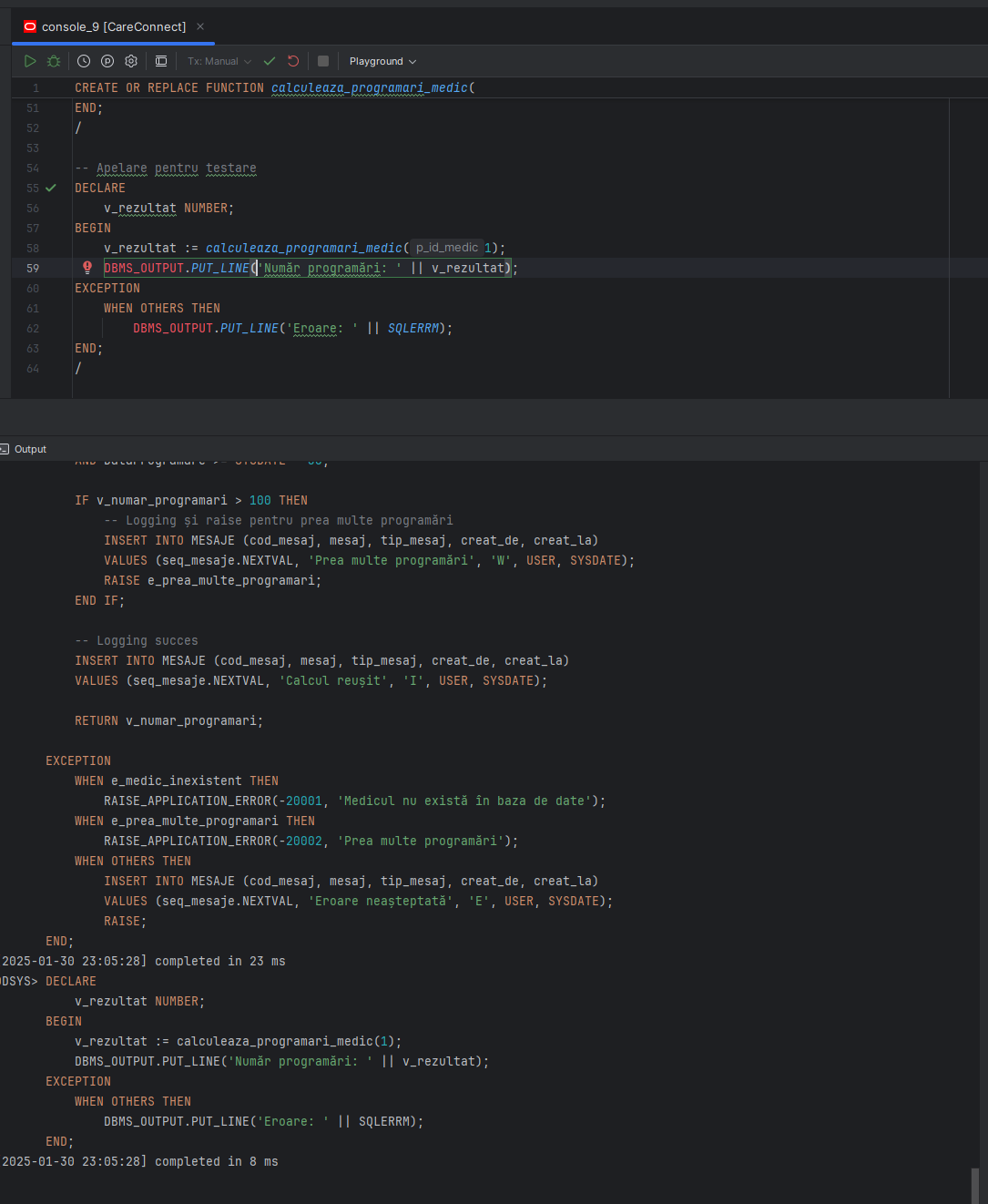
EXCEPTION

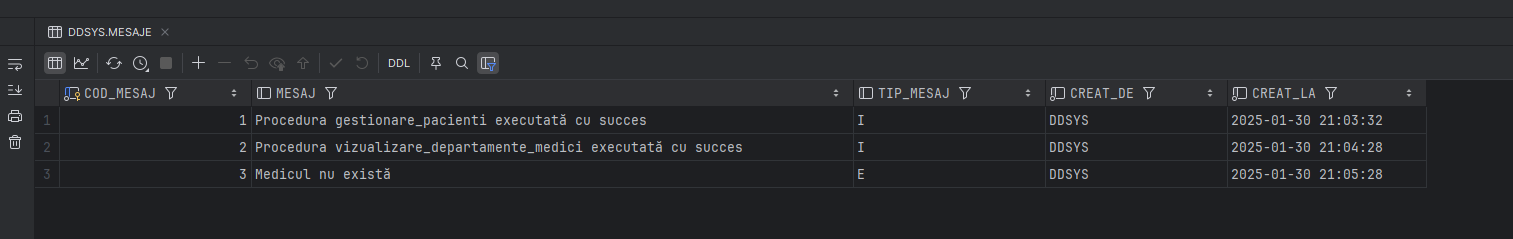
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: ' || SQLERRM);

END;

/





Punct 4

CREATE OR REPLACE TRIGGER trg\_verificare\_programari

BEFORE INSERT OR UPDATE ON Programare

BEGIN

-- Verifică dacă se încearcă programări în weekend

IF TO\_CHAR(SYSDATE, 'DY') IN ('SAT', 'SUN') THEN

-- Logging avertisment

INSERT INTO MESAJE (cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)

VALUES (seq\_mesaje.NEXTVAL,

'Nu se pot face programări în weekend',

'W',

USER,

SYSDATE);

RAISE\_APPLICATION\_ERROR(-20003, 'Nu se pot face programări în weekend');

END IF;

-- Logging succes

INSERT INTO MESAJE (cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)

VALUES (seq\_mesaje.NEXTVAL,

'Verificare programări efectuată cu succes',

'I',

USER,

SYSDATE);

END;

/

-- Test trigger

BEGIN

-- Încercare de inserare programare

INSERT INTO Programare (ID, IDPacient, IDMedic, DataProgramare)

VALUES (SEQ\_PROGRAMARE\_ID.NEXTVAL, 1, 1, SYSDATE);

COMMIT;

EXCEPTION

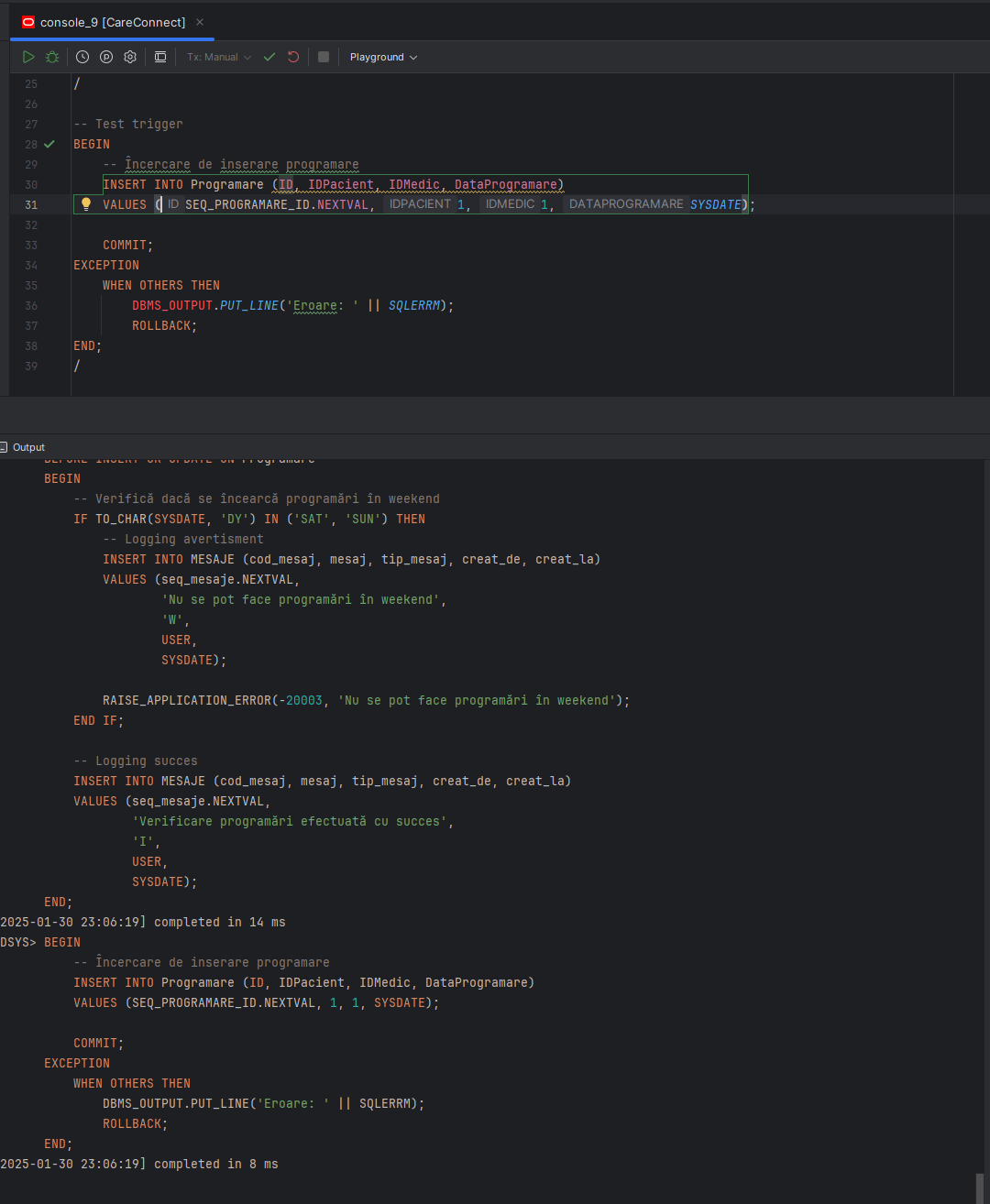
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: ' || SQLERRM);

ROLLBACK;

END;

/



Punct 5

CREATE OR REPLACE TRIGGER trg\_verificare\_medic

BEFORE INSERT OR UPDATE ON Medic

FOR EACH ROW

BEGIN

-- Verifică dacă gradul profesional este valid

IF :NEW.GradProfesional NOT IN ('Rezident', 'Specialist', 'Primar') THEN

-- Logging eroare

INSERT INTO MESAJE (cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)

VALUES (

seq\_mesaje.NEXTVAL,

'Grad profesional invalid: ' || :NEW.GradProfesional,

'E',

USER,

SYSDATE

);

RAISE\_APPLICATION\_ERROR(-20004, 'Grad profesional invalid');

END IF;

-- Logging modificare

INSERT INTO MESAJE (cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)

VALUES (

seq\_mesaje.NEXTVAL,

'Modificare medic: ' || :NEW.Nume,

'I',

USER,

SYSDATE

);

END;

/

-- Test trigger

BEGIN

-- Încercare de inserare medic cu grad invalid

INSERT INTO Medic (ID, Nume, Prenume, GradProfesional, IDDepartament)

VALUES (SEQ\_MEDIC\_ID.NEXTVAL, 'Test', 'Test', 'Invalid', 1);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: ' || SQLERRM);

ROLLBACK;

END;

/

PUNCT 6

CREATE OR REPLACE TRIGGER trg\_monitorizare\_schema

AFTER DDL ON SCHEMA

BEGIN

-- Logging modificare structură

INSERT INTO MESAJE (

cod\_mesaj,

mesaj,

tip\_mesaj,

creat\_de,

creat\_la

)

VALUES (

seq\_mesaje.NEXTVAL,

'Operație DDL executată: ' || ora\_sysevent,

'I',

user,

SYSDATE

);

END;

/

-- Test trigger

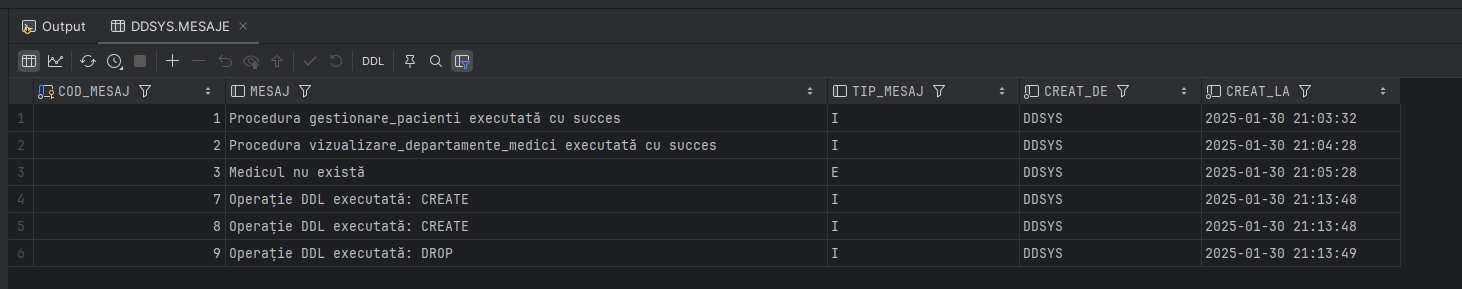
CREATE TABLE test\_table (

id NUMBER PRIMARY KEY,

nume VARCHAR2(100)

);

DROP TABLE test\_table;



Punct 7

CREATE OR REPLACE PACKAGE management\_spital AS

-- Excepții

e\_medic\_inexistent EXCEPTION;

e\_prea\_multe\_programari EXCEPTION;

-- Procedura cu colecții

PROCEDURE gestionare\_pacienti;

-- Procedura cu cursoare

PROCEDURE vizualizare\_departamente\_medici;

-- Funcția cu 3 tabele

FUNCTION calculeaza\_programari\_medic(p\_id\_medic NUMBER) RETURN NUMBER;

-- Proceduri pentru testarea trigger-urilor

PROCEDURE test\_inserare\_programare(

p\_id\_pacient NUMBER,

p\_id\_medic NUMBER,

p\_data DATE

);

PROCEDURE test\_inserare\_medic(

p\_nume VARCHAR2,

p\_prenume VARCHAR2,

p\_grad VARCHAR2,

p\_id\_dept NUMBER

);

END management\_spital;

/

CREATE OR REPLACE PACKAGE BODY management\_spital AS

-- Implementare procedură cu colecții

PROCEDURE gestionare\_pacienti IS

TYPE t\_pacienti IS TABLE OF Pacient%ROWTYPE INDEX BY PLS\_INTEGER;

v\_pacienti t\_pacienti;

BEGIN

SELECT \* BULK COLLECT INTO v\_pacienti

FROM Pacient

WHERE ROWNUM <= 5;

INSERT INTO MESAJE (cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)

VALUES (seq\_mesaje.NEXTVAL, 'Gestionare pacienți executată', 'I', USER, SYSDATE);

END gestionare\_pacienti;

-- Implementare procedură cu cursoare

PROCEDURE vizualizare\_departamente\_medici IS

CURSOR c\_departamente IS

SELECT id, NumeDepartament FROM Departament;

CURSOR c\_medici (p\_id\_dept NUMBER) IS

SELECT Nume, Prenume FROM Medic WHERE IDDepartament = p\_id\_dept;

BEGIN

FOR dep IN c\_departamente LOOP

FOR med IN c\_medici(dep.id) LOOP

NULL; -- Aici doar pentru demonstrație

END LOOP;

END LOOP;

INSERT INTO MESAJE (cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)

VALUES (seq\_mesaje.NEXTVAL, 'Vizualizare departamente executată', 'I', USER, SYSDATE);

END vizualizare\_departamente\_medici;

-- Implementare funcție

FUNCTION calculeaza\_programari\_medic(p\_id\_medic NUMBER) RETURN NUMBER IS

v\_numar\_programari NUMBER := 0;

v\_exista\_medic NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_exista\_medic FROM Medic WHERE ID = p\_id\_medic;

IF v\_exista\_medic = 0 THEN

RAISE e\_medic\_inexistent;

END IF;

SELECT COUNT(\*) INTO v\_numar\_programari

FROM Programare

WHERE IDMedic = p\_id\_medic;

RETURN v\_numar\_programari;

END calculeaza\_programari\_medic;

-- Implementare procedură test programare

PROCEDURE test\_inserare\_programare(

p\_id\_pacient NUMBER,

p\_id\_medic NUMBER,

p\_data DATE

) IS

BEGIN

INSERT INTO Programare (ID, IDPacient, IDMedic, DataProgramare)

VALUES (SEQ\_PROGRAMARE\_ID.NEXTVAL, p\_id\_pacient, p\_id\_medic, p\_data);

INSERT INTO MESAJE (cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)

VALUES (seq\_mesaje.NEXTVAL, 'Programare inserată', 'I', USER, SYSDATE);

END test\_inserare\_programare;

-- Implementare procedură test medic

PROCEDURE test\_inserare\_medic(

p\_nume VARCHAR2,

p\_prenume VARCHAR2,

p\_grad VARCHAR2,

p\_id\_dept NUMBER

) IS

BEGIN

INSERT INTO Medic (ID, Nume, Prenume, GradProfesional, IDDepartament)

VALUES (SEQ\_MEDIC\_ID.NEXTVAL, p\_nume, p\_prenume, p\_grad, p\_id\_dept);

INSERT INTO MESAJE (cod\_mesaj, mesaj, tip\_mesaj, creat\_de, creat\_la)

VALUES (seq\_mesaje.NEXTVAL, 'Medic inserat', 'I', USER, SYSDATE);

END test\_inserare\_medic;

END management\_spital;

/

-- Test pachet

BEGIN

-- Test procedură cu colecții

management\_spital.gestionare\_pacienti;

-- Test procedură cu cursoare

management\_spital.vizualizare\_departamente\_medici;

-- Test funcție

DBMS\_OUTPUT.PUT\_LINE('Număr programări: ' ||

management\_spital.calculeaza\_programari\_medic(1));

-- Test inserare programare

management\_spital.test\_inserare\_programare(1, 1, SYSDATE);

-- Test inserare medic

management\_spital.test\_inserare\_medic('Test', 'Test', 'Specialist', 1);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: ' || SQLERRM);

END;

/

