

# **Proiect**

## **Data Warehouse & Business Intelligence**

### **Echipa 4:**

*Dobre Mihaela Beatrice*

*Mocanu Alina Cristina*

*Muşat Andreea*

*Surcea Mihai Daniel*

# RAPORTUL DE ANALIZĂ

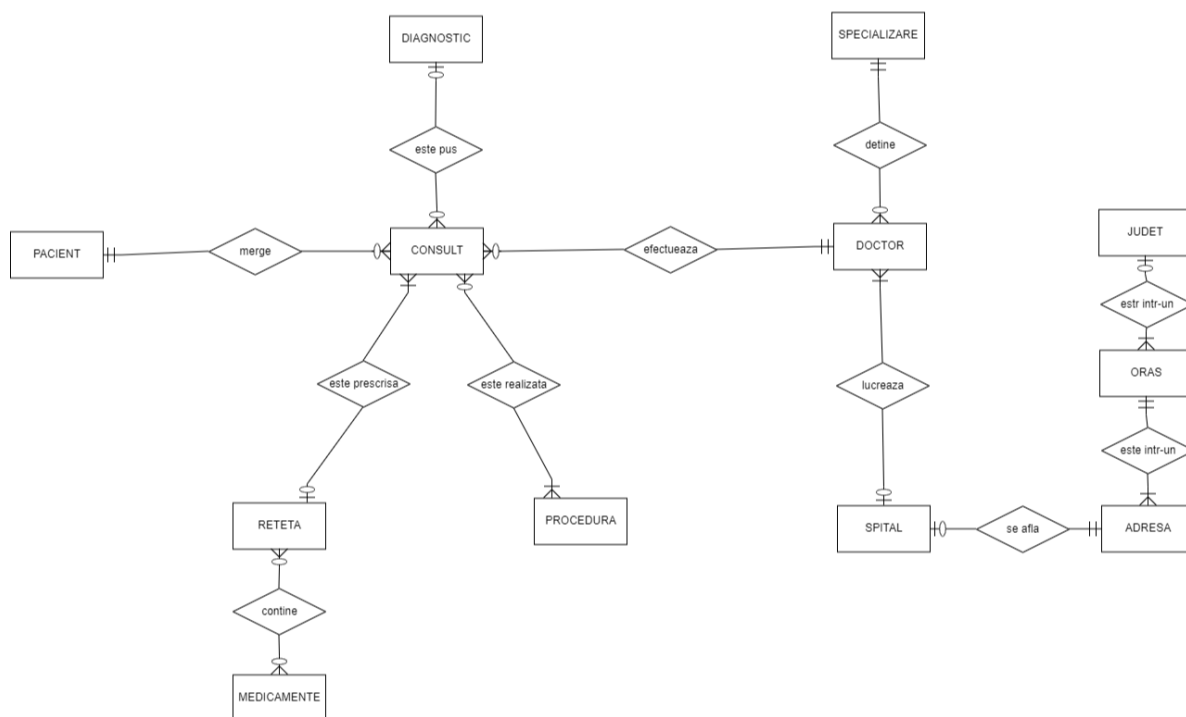
## 1. Descrierea modelului ales și a obiectivelor aplicației

În cadrul proiectului a fost implementată o bază de date OLTP care ajută la gestionarea consulturilor medicale ale pacienților în spitalele private din diferite orașe ale țării. Un pacient este consultat de către un doctor specializat dintr-un spital, iar prin intermediul uneia sau a mai multor proceduri medicale efectuate acesta primește un diagnostic. De asemenea, pacientul primește la finalul consultației și o rețetă cu medicamente în caz de nevoie.

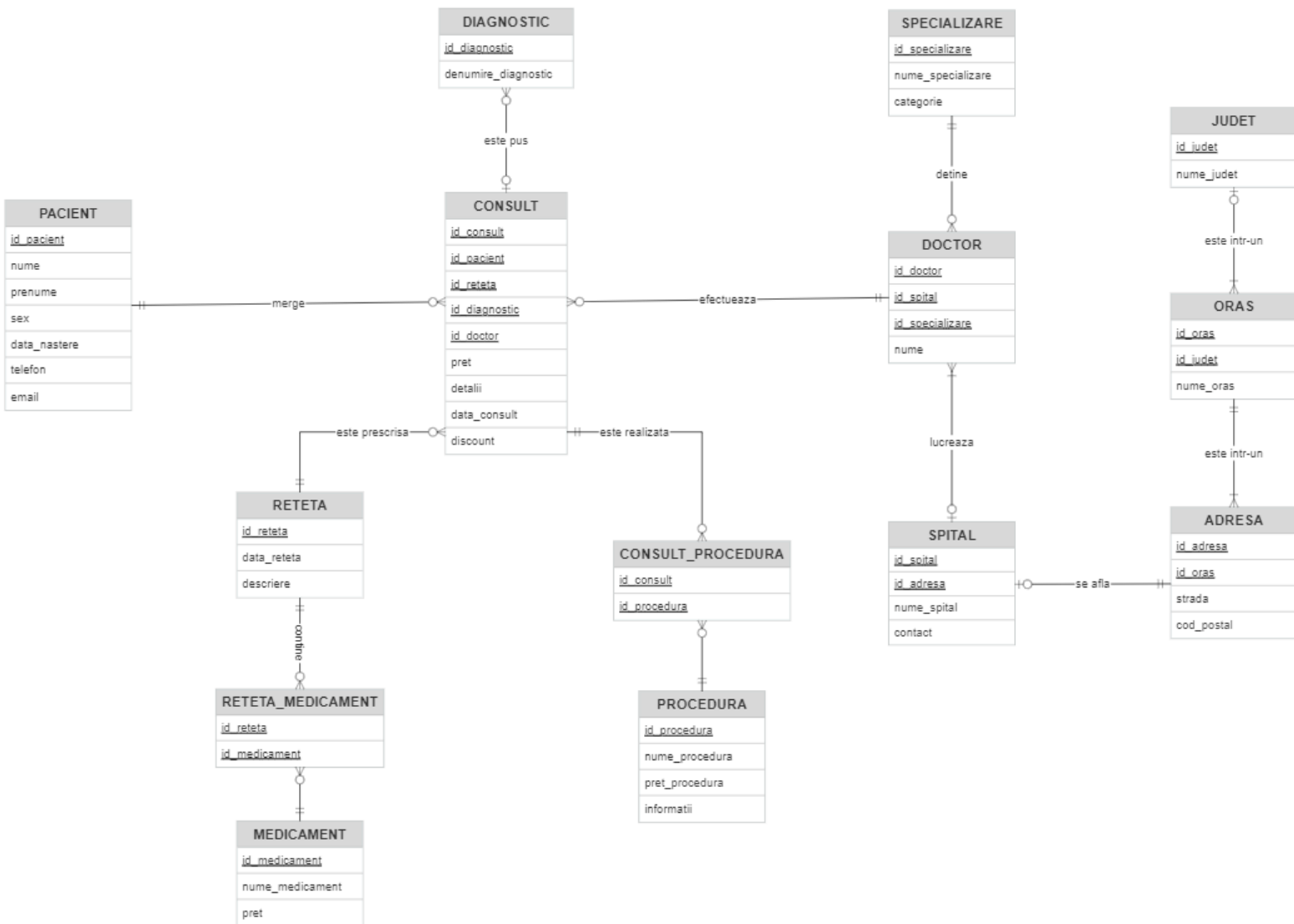
Data Warehouse-ul realizat pe baza acestei baze de date se focusează pe colectarea și analizarea de date pentru întocmirea unor statistici privind costurile pentru pacienți necesare desfășurării acestor activități în diferite perioade de timp, precum prețul vizitelor la doctor care includ consultațiile(inclusiv procedurile efectuate în cadrul acestora) și medicamentele din rețetele eliberate în funcție de diagnostic, spital sau doctor. Se va construi depozitul de date astfel încât să conțină toate datele necesare pentru costuri și asigurând integrarea, persistența și caracterul istoric al datelor.

## 2. Diagramele bazei de date OLTP

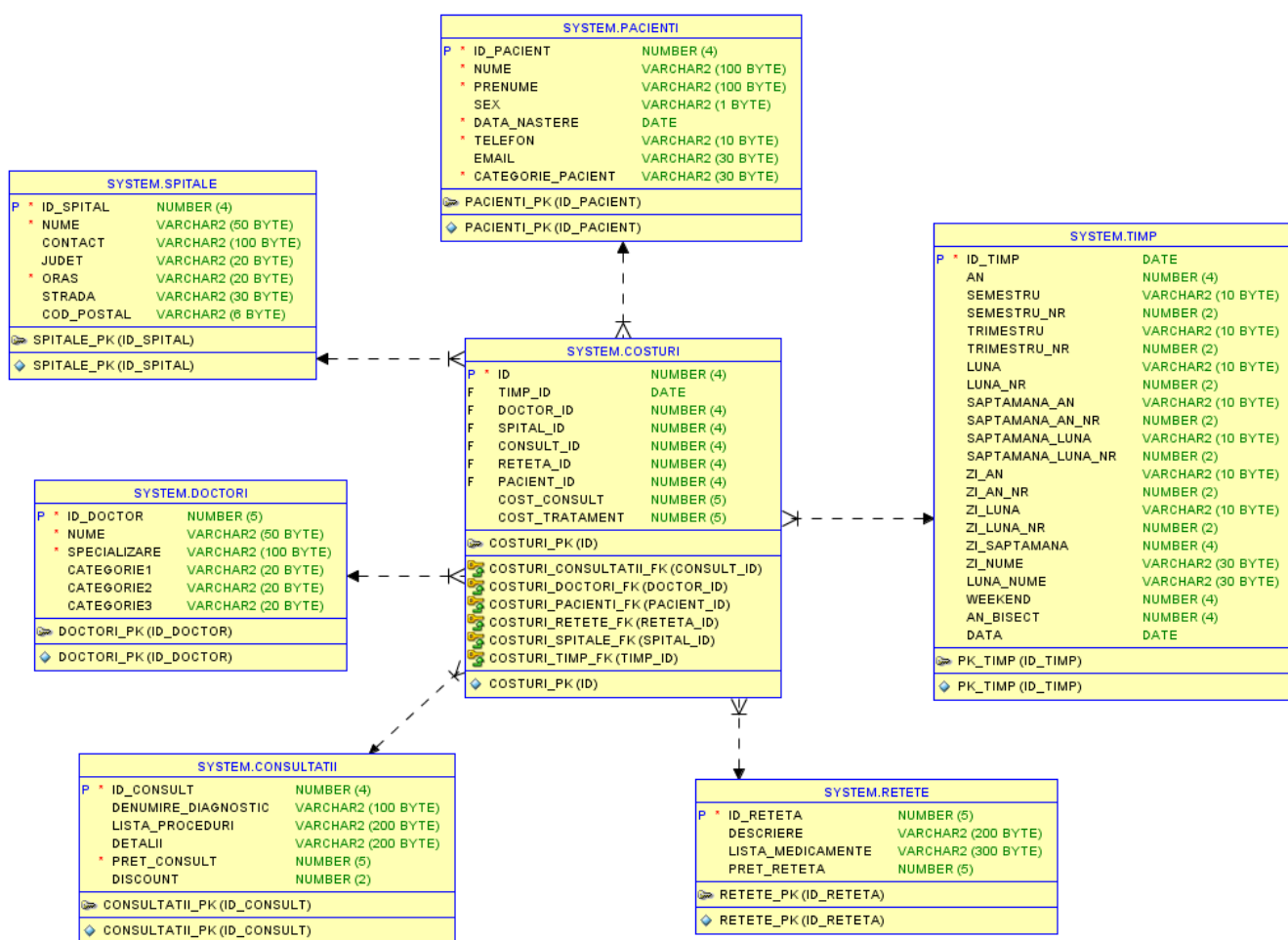
### a) Diagrama entitate – relație a bazei de date OLTP



**b) Diagrama conceptuală a bazei de date OLTP**



### 3. Diagrama stea/fulg a bazei de date depozit



### 4. Descrierea câmpurilor necesare pentru fiecare tabel din baza de date depozit și modul de populare al acestora cu informații din baza de date OLTP

Informațiile din depozitul de date vor fi colectate și actualizate prin intermediul unui proces ETL(Extract, Transform, Load). Datele vor fi extrase din cele 14 entități ale bazei de date OLTP, vor fi prelucrate atunci când este cazul, iar mai apoi vor fi încărcate în depozit. Acest proces va avea loc săptămânal.

Tabela dimensiune PACIENȚI:

- id\_pacient: număr, preluat din OLTP din tabelul PACIENT
- nume: șir de caractere, preluat din OLTP din tabelul PACIENT
- prenume: șir de caractere, preluat din OLTP din tabelul PACIENT
- sex: șir de caractere, preluat din OLTP din tabelul PACIENT
- data\_nastere: dată, preluată din OLTP din tabelul PACIENT

- telefon: șir de caractere, preluat din OLTP din tabelul PACIENT
- email: șir de caractere, preluat din OLTP din tabelul PACIENT
- categorie\_pacient: șir de caractere, calculat în funcție de vârsta pacientului: copil dacă este sub 18 ani, adult dacă este între 18 și 61 de ani la femei, respectiv între 18 și 65 de ani la bărbați, și pensionar peste varstele menționate.

#### Tabela dimensiune SPITALE:

- id\_spital: număr, preluat din OLTP din tabelul SPITAL
- nume: șir de caractere, preluat din OLTP din tabelul SPITAL
- contact: șir de caractere, preluat din OLTP din tabelul SPITAL, reprezentat de un număr de telefon sau o adresă de email
- județ: șir de caractere, preluat din OLTP prin JOIN-ul tabelelor SPITAL, ADRESĂ, ORAȘ și JUDEȚ
- oraș: șir de caractere, preluat din OLTP prin JOIN-ul tabelelor SPITAL, ADRESĂ și ORAȘ
- stradă: șir de caractere, preluat din OLTP prin JOIN-ul tabelelor SPITAL și ADRESĂ
- cod\_poștal: șir de caractere, preluat din OLTP prin JOIN-ul tabelelor SPITAL și ADRESĂ

#### Tabela dimensiune DOCTORI:

- id\_doctor: număr, preluat din OLTP din tabelul DOCTOR
- nume: șir de caractere, preluat din OLTP din tabelul DOCTOR
- specializare: șir de caractere, denumirea specializării doctorului, regăsită în OLTP prin JOIN-ul tabelelor DOCTOR și SPECIALIZARE + SELF JOIN-ul tablei SPECIALIZARE
- categorie1: șir de caractere, regăsit în OLTP prin JOIN-ul tabelelor DOCTOR și SPECIALIZARE + SELF JOIN-ul tablei SPECIALIZARE, reprezentând titulatura unui doctor(medic rezident, specialist, primar)
- categorie2: șir de caractere, regăsit în OLTP prin JOIN-ul tabelelor DOCTOR și SPECIALIZARE + SELF JOIN-ul tablei SPECIALIZARE, reprezentând subdomeniul în care activează
- categorie3: șir de caractere, regăsit în OLTP prin JOIN-ul tabelelor DOCTOR și SPECIALIZARE + SELF JOIN-ul tablei SPECIALIZARE, reprezentând domeniul general în care este specializat

Tabela dimensiune CONSULTAȚII:

- id\_consult: număr, preluat din OLTP din tabelul CONSULT
- denumire\_diagnostic: șir de caractere, preluat din OLTP prin JOIN-ul tabelelor CONSULT și DIAGNOSTIC
- lista\_proceduri: șir de caractere, obținut prin concatenarea tuturor numelor procedurilor medicale efectuate la un consult prin JOIN-UL tabelelor CONSULT\_PROCEDURA și PROCEDURĂ din OLTP
- detalii: șir de caractere, preluat din OLTP din tabelul CONSULT
- pret\_consult: număr, preluat din OLTP din tabelul CONSULT
- discount: număr ce servește drept procent, preluat din OLTP din tabelul CONSULT

Tabela dimensiune REȚETE:

- id\_rețetă: număr, preluat din OLTP din tabelul REȚETĂ
- descriere: șir de caractere, preluat din OLTP din tabelul REȚETĂ
- listă\_medicamente: șir de caractere, obținut prin concatenarea tuturor medicamentelor prescrise într-o rețetă prin JOIN-ul tabelelor REȚETĂ\_MEDICAMENT și MEDICAMENT
- preț\_rețetă: șir de caractere, obținut prin adunarea tuturor prețurilor medicamentelor prescrise într-o rețetă prin JOIN-ul tabelelor REȚETĂ\_MEDICAMENT și MEDICAMENT

Tabela dimensiune TIMP: id-ul este preluat din OLTP, reprezentând *data\_consult* din tabela CONSULT, restul coloanelor fiind derivate din această dată, calculate folosind funcții de manipulare a datelor din cadrul SQL.

Tabela de fapte COSTURI:

- id: număr, generat automat incremental
- timp\_id: cheie externă către tabela dimensiune TIMP
- doctor\_id: cheie externă către tabela dimensiune DOCTORI
- spital\_id: cheie externă către tabela dimensiune SPITALE
- consult\_id: cheie externă către tabela dimensiune CONSULTAȚII
- reteta\_id: cheie externă către tabela dimensiune REȚETE
- pacient\_id: cheie externă către tabela dimensiune PACIENȚI

- cost\_consult: prețul unui consult calculat prin prețul consultației propriu zise + prețul procedurilor efectuate - discountul oferit la consult.
- cost\_tratament: prețul unei scheme de tratament care reprezintă prețul medicamentelor de pe o rețetă.

## 5. *Identificarea constrângerilor specifice depozitelor de date ce trebuie definite*

Pentru constrângerile care conțin opțiunea **RELY**, aceasta a fost aleasă deoarece datele provin direct din baza de date OLTP, fără alte transformări, unde au fost validate pe aceleași criterii, iar o nouă validare nu mai este necesară, pentru că ne dorim să evităm consumul inutil de resurse.

Constrângeri pentru tabela PACIENȚI:

- PRIMARY KEY RELY *id\_pacient* pentru identificarea unică a pacientului.
- NOT NULL RELY pentru *nume, prenume, data\_naștere, telefon* pentru a ne asigura că putem identifica corect pacientul și avem toate informațiile personale necesare pentru acesta.
- CHECK (LENGTHB(telefon) = 10) ENABLE NOVALIDATE pentru a fi un număr de telefon valid în România. Această constrângere trebuie respectată pentru datele care urmează a fi încărcate, însă nu se aplică și pentru cele deja existente, deoarece este posibil să existe numere cu prefix în față(+40) sau cetățeni români cu numere străine, iar acest lucru nu ne afectează în realizarea rapoartelor.
- CHECK (sex ='F' OR sex = 'M') ENABLE VALIDATE pentru a avea un mod unic de a reprezenta sexul pacienților. Opțiunea ENABLE VALIDATE a fost aleasă pentru a ne asigura ca datele deja existe în data warehouse respecta acest format și că datele care urmează a fi încărcate sunt, de asemenea, corecte, deoarece vom avea nevoie de același format pentru realizarea rapoartelor.

Constrângeri pentru tabela SPITALE:

- PRIMARY KEY RELY *id\_spital* pentru identificarea unică a spitalului.
- NOT NULL RELY pentru numele spitalului și orașul în care este situat, deoarece reprezintă informații esențiale în analiza și statisticile datelor.

Constrângeri pentru tabela DOCTORI:

- PRIMARY KEY RELY *id\_doctor* pentru identificarea unică a doctorului.

- NOT NULL RELY pentru numele doctorului și a specializării sale, deoarece reprezintă informații esențiale în analiza și statisticile datelor.

Constrângeri pentru tabela RETETE:

- PRIMARY KEY RELY *id\_reteta* pentru identificarea unică a retetei.

Constrângeri pentru tabela CONSULTAȚII:

- PRIMARY KEY RELY *id\_consult* pentru identificarea unică a consultului.
- NOT NULL RELY pentru prețul consultului, deoarece reprezintă informații esențiale în analiza și statisticile datelor.
- CHECK RELY pentru *discount* pentru a ne "aminti" care sunt valorile permise ale acestei coloane, deși constrângerea a fost verificată în adăugarea în baza de date.

Constrângeri pentru tabela TIMP:

- PRIMARY KEY ENABLE VALIDATE *id\_timp* pentru identificarea unică a înregistrării. Opțiunea ENABLE VALIDATE a fost aleasă inițial pentru a ne asigura în orice moment că nu există date duplicate sau nule în tabelă.

Constrângeri pentru tabela de fapte COSTURI:

- PRIMARY KEY ENABLE VALIDATE *id\_cost* pentru identificarea unică a înregistrării. Opțiunea ENABLE VALIDATE a fost aleasă pentru a ne asigura în orice moment că nu există date duplicate sau nule în tabelă.
- FOREIGN KEY DISABLE NOVALIDATE către *id\_timp*, deoarece acesta nu implică validarea datelor existente în cele două tabele și nici verificări ulterioare în cazul comenzilor LMD executate pe acestea, iar optimizatorul va putea utiliza constrângerea pentru a determina un plan optim.
- FOREIGN KEY ENABLE NO VALIDATE cu id-urile din tabelele DOCTORI, SPITALE, CONSULTATII, RETETE, PACIENTI deoarece procesul ETL verifică îndeplinirea constrângerii de cheie externă, dar totuși se dorește menținerea sa și la nivelul depozitului de date pentru a preveni orice modificări ce pot afecta constrângerea de cheie externă în afara procesului ETL.



**6. Identificarea indecșilor specifici depozitelor de date ce trebuie definiți asupra modelului; formularea unei cereri în limbaj natural care va determina utilizarea indecșilor specificați și va fi implementată în următoarea etapă**

Indecșii specifici depozitelor de date sunt cei de tipul Bitmap. Vor fi definiți doi indecși de tip bitmap asupra a două tabele dimensiune ale modelului:

- Coloana *categorie2* a tabeli dimensiune DOCTORI va fi indexată, deoarece reprezintă titulatura doctorului: rezident, specialist sau primar.
- Coloana *categorie\_pacient* a tabeli dimensiune PACIENT va fi indexată, deoarece poate fi copil, adult sau pensionar.
- Coloana *discount* a tabeli CONSULT va fi indexată, deoarece poate fi 0%, 25%, 50%, 75%.

Aceste două coloane pot conține valori care nu sunt distincte pentru fiecare înregistrare și au cardinalitate mică, fiind buni candidați pentru crearea de indecși bitmap.

Cerere: Să se afișeze costul total al consultațiilor pe care le-au avut pensionarii în ultima lună a anului 2021, numărul de doctori rezidenți din țară și prețul mediu al unui consult în care s-a acordat un discount mai mare de 25%.

**7. Identificarea obiectelor de tip dimensiune ce trebuie definite asupra modelului**

În cadrul Data Warehouse-ului am identificat 2 obiecte de tip dimensiune și implicit două ierarhii.

Prima este cea din tabelul SPITALE pentru adresa la care se află acesta:

spital → cod poștal → strada → oraș → județ

Exemplu:

MedCare(denumirea spitalului)

1000345 (codul poștal de pe strada, din orașul și din județul unde se află spitalul)

Splaiul Independenței(strada unde se află spitalul)

București(orașul unde se află spitalul)

București(județul unde se află spitalul)

De asemenea, se poate observa că unei valori din nivelul ierarhic *spital* îi corespund în mod unic valori ale atributelor *nume* și *contact*.

A doua dimensiune identificată este cea din tabelul DOCTORI, pentru specializarea, calificarea și domeniul de activitate al unui doctor.

doctor → specializare → titulatura → subdomeniu → domeniu

Exemplu:

Zaharescu Iulian(numele doctorului)

Chirurg(numele specializării sale)

Medic specialist(gradul, în funcție de examenele luate - poate fi rezident, specialist sau primar)

Chirurgie endocrină(subdomeniul)

Chirurgie oncologică(domeniul)

***8. Identificarea tabelelor care vor fi partiționate și a tipului de partiționare (minim 2 dacă echipa este formată din 4 persoane); formularea unei cereri în limbaj natural care va determina utilizarea lor și va fi implementată în următoarea etapă***

Prima tabelă partiționată va fi tabela de fapte, după timp, pentru a avea o evidență mai clară a evoluției costurilor. În tabela de fapte vor fi introduse date despre consulturi începând cu data de 1 ianuarie 2022.

A doua tabelă selectată pentru partiționare este cea unde vor fi stocate informații despre spitale, pentru a putea identifica mai ușor spitalele în funcție de regiunea țării. (Moldova, Transilvania, Oltenia, Muntenia, Dobrogea).

Rata de îmbolnăvire de gripă în funcție de zonele țării în ultimele 3 luni ale anului trecut.

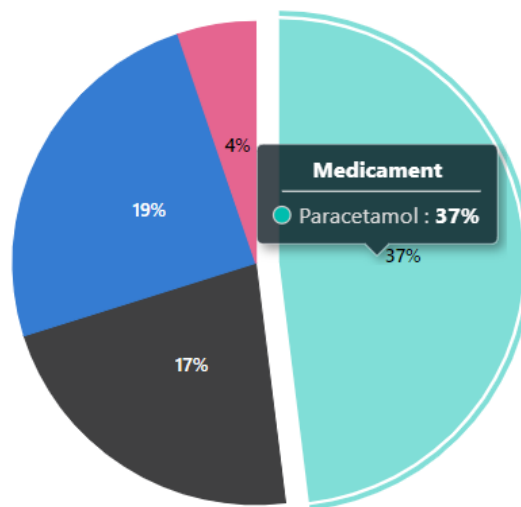
***9. Formularea în limbaj natural a unei cereri SQL complexe care va fi optimizată în următoarea etapă, folosind tehnici specifice bazelor de date depozit. Precizarea tehnicilor de optimizare ce ar putea fi utilizate pentru această cerere particulară (avantaje / dezavantaje de utilizare pentru o anumită tehnică)***

Afișați evoluția săptămânală a consumului de Nurofen la copii din trimestrul 2 al anului trecut și costul mediu al tratamentelor lor care au Nurofen în rețetă.

**10. Formularea în limbaj natural a cel puțin 5 cereri cu grad de complexitate diferită, concretizate în rapoarte (grafice) ce vor fi create în următoarele etape**

1. Afișați procentele medicamentelor vândute (raceala - Paracetamol, antiinflamator - Diclofenac, durere - Voltaren, stomac - No-spa) în anul în care s-a înregistrat cel mai mare cost al unui tratament (în caz de egalitate, se va lua cel mai recent an).

Procentele medicamentelor vândute

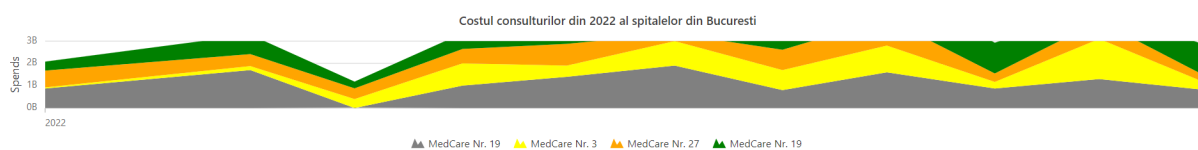


2. Să se afișeze evoluția mediilor lunare a costurilor pentru tratamentul pacienților care au fost diagnosticați cu cancer din anul 2022 (anul reprezintă durata pe care se calculează mediile, nu perioada diagnosticării).

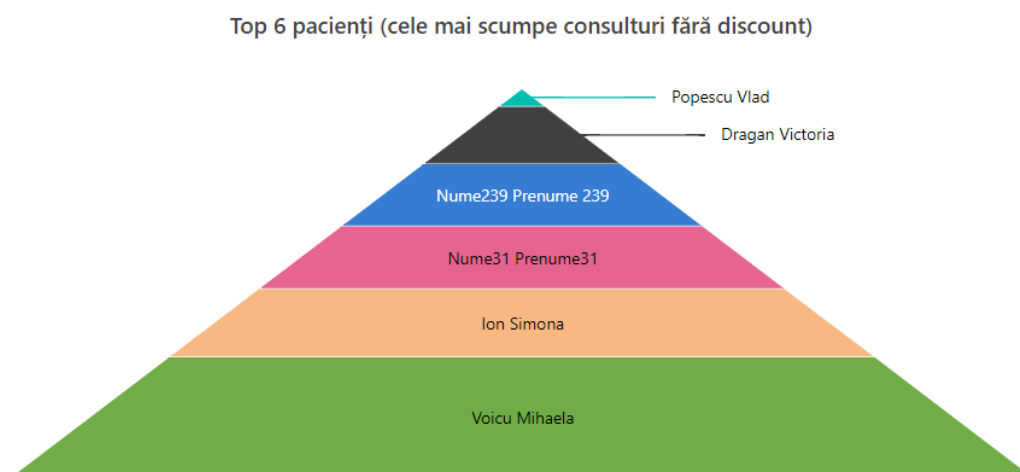
Evoluția costurilor pentru tratamentul de cancer - anul 2022



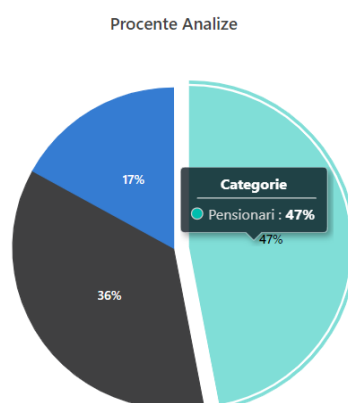
- Să se obțină valoarea costului consulturilor din anul 2022 al fiecărui spital din București.



- Să se obțină numele de la top 6 pacienți (separați pe categorie de sex) care au avut cele mai scumpe consulturi fără discount.



- Să se determine numerele din fiecare categorie de vârstă al pacienților care și-au făcut analizele la spitalele din București (se ia în considerare vârsta la momentul recoltării).



# MODUL IMPLEMENTARE BAZE DE DATE

## 1. Crearea bazei de date OLTP și a utilizatorilor

```
DROP TABLE CONSULT_PROCEDURA;
DROP TABLE RETETA_MEDICAMENT;
DROP TABLE MEDICAMENT;
ALTER TABLE CONSULT DROP CONSTRAINT CONSULT_fk_RETETA;
DROP TABLE RETETA;
DROP TABLE CONSULT;
DROP TABLE PACIENT;
DROP TABLE DOCTOR;
DROP TABLE PROCEDURA;
DROP TABLE SPECIALIZARE;
DROP TABLE DIAGNOSTIC;
DROP TABLE SPITAL;
DROP TABLE ADRESA;
DROP TABLE ORAS;
DROP TABLE JUDET;

--Crearea tabelului JUDET:
CREATE TABLE JUDET(
    id_judet NUMBER(4) GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by 1),
    nume_judet VARCHAR(20) NOT NULL,
    CONSTRAINT judet_pk PRIMARY KEY (id_judet)
);

--CREAREA TABELULUI ORAS:
CREATE TABLE ORAS(
    id_oras NUMBER(4) GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by 1),
    id_judet Number(4),
    nume_oras VARCHAR2(20) NOT NULL,
    CONSTRAINT oras_pk PRIMARY KEY (id_oras),
    CONSTRAINT oras_fk_judet FOREIGN KEY(id_judet) REFERENCES JUDET(id_judet) ON
DELETE CASCADE
);

--Crearea tabelului ADRESA:
CREATE TABLE ADRESA(
    id_adresa NUMBER(4) GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by 1),
    id_oras NUMBER(4),
    strada VARCHAR2(30),
    cod_postal VARCHAR2(6),
    CONSTRAINT adresa_pk PRIMARY KEY (id_adresa),
    CONSTRAINT adresa_fk_oras FOREIGN KEY(id_oras) REFERENCES ORAS(id_oras) ON
DELETE CASCADE
);
```

--Crearea tabelului PACIENT:

```
CREATE TABLE PACIENT(  
    id_pacient NUMBER(4) GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by 1),  
    nume VARCHAR2(30) NOT NULL,  
    prenume VARCHAR2(50) NOT NULL,  
    sex VARCHAR2(1),  
    data_nastere DATE NOT NULL,  
    telefon VARCHAR2(100) NOT NULL,  
    email VARCHAR2(100),  
    CONSTRAINT pacient_pk PRIMARY KEY (id_pacient),  
    CONSTRAINT pacient_phone CHECK (LENGTHB(telefon) = 10)  
);
```

--Crearea tabelului SPITAL

```
CREATE TABLE SPITAL(  
    id_spital NUMBER(4) GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by 1),  
    id_adresa NUMBER(4),  
    nume_spital VARCHAR2(50) NOT NULL,  
    contact VARCHAR2(100),  
    CONSTRAINT spital_pk PRIMARY KEY (id_spital),  
    CONSTRAINT spital_fk_adr FOREIGN KEY(id_adresa) REFERENCES ADRESA(id_adresa) ON  
DELETE CASCADE  
);
```

--Crearea tabelului MEDICAMENT

```
CREATE TABLE MEDICAMENT(  
    id_medicament NUMBER(4) GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT  
by 1),  
    nume_medicament VARCHAR2(50) NOT NULL,  
    pret NUMBER(4,2) DEFAULT 0,  
    CONSTRAINT medicament_pk PRIMARY KEY (id_medicament)  
);
```

--Crearea tabelului SPECIALIZARE

```
CREATE TABLE SPECIALIZARE(  
    id_specializare NUMBER(4), -- GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT  
by 1),  
    nume_specializare VARCHAR2(100) NOT NULL,  
    categorii number(4),  
    CONSTRAINT specializare_pk PRIMARY KEY (id_specializare)  
);
```

--Crearea tabelului DOCTOR

```
CREATE TABLE DOCTOR(  
    id_doctor NUMBER(4) GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by 1),  
    id_spital NUMBER(4),
```

```

id_specializare NUMBER(4),
nume VARCHAR2(50) NOT NULL,
data_angajare DATE,
CONSTRAINT doctor_pk PRIMARY KEY (id_doctor),
CONSTRAINT doctor_spital_fk FOREIGN KEY (id_spital) REFERENCES SPITAL(id_spital) ON
DELETE CASCADE,
CONSTRAINT doctor_specializare_fk FOREIGN KEY (id_specializare) REFERENCES
SPECIALIZARE(id_specializare) ON DELETE CASCADE
);

--Crearea tabelului DIAGNOSTIC
CREATE TABLE DIAGNOSTIC(
    id_diagnostic NUMBER(4) GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by
1),
denumire_diagnostic VARCHAR2(100),
CONSTRAINT diagnostic_pk PRIMARY KEY (id_diagnostic)
);

--Crearea tabelului RETETA
CREATE TABLE RETETA(
    id_reteta NUMBER(4) GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by 1),
    data_reteta DATE DEFAULT SYSDATE,
    descriere VARCHAR2(200),
    CONSTRAINT reteta_pk PRIMARY KEY (id_reteta)
);

--Crearea tabelului RETETA_MEDICAMENT:
CREATE TABLE RETETA_MEDICAMENT(
    id_reteta NUMBER(4),
    id_medicament NUMBER(4),
    CONSTRAINT reteta_med_pk_ PRIMARY KEY(id_reteta, id_medicament),
    CONSTRAINT medicament_fk FOREIGN KEY (id_medicament) REFERENCES
MEDICAMENT(id_medicament) ON DELETE CASCADE,
    CONSTRAINT reteta_fk FOREIGN KEY (id_reteta) REFERENCES RETETA(id_reteta) ON
DELETE CASCADE
);

--Crearea tabelului PROCEDURA:
CREATE TABLE PROCEDURA(
    id_procedura NUMBER(5) GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by
1),
    nume_procedura VARCHAR2(100) NOT NULL,
    pret_procedura NUMBER(5) DEFAULT 0 NOT NULL,
    informatii VARCHAR2(200),
    CONSTRAINT procedura_pk PRIMARY KEY (id_procedura)
);

```

--Crearea tabelului CONSULT:

```
CREATE TABLE CONSULT(  
    id_consult NUMBER(4) GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by 1),  
    id_pacient NUMBER(4),  
    id_reteta NUMBER(4),  
    id_diagnostic NUMBER(4),  
    id_doctor NUMBER(4),  
    pret NUMBER(5),  
    data_consult DATE NOT NULL,  
    detalii VARCHAR2(200),  
    discount NUMBER(3) DEFAULT 0,  
    CONSTRAINT consult_pk PRIMARY KEY (id_consult),  
    CONSTRAINT consult_fk_pacient FOREIGN KEY (id_pacient) REFERENCES  
PACIENT(id_pacient) ON DELETE CASCADE,  
    CONSTRAINT consult_fk_doctor FOREIGN KEY (id_doctor) REFERENCES DOCTOR(id_doctor)  
ON DELETE CASCADE,  
    CONSTRAINT consult_fk_reteta FOREIGN KEY (id_reteta) REFERENCES RETETA(id_reteta) ON  
DELETE CASCADE,  
    CONSTRAINT consult_fk_diagnostic FOREIGN KEY (id_diagnostic) REFERENCES  
DIAGNOSTIC(id_diagnostic) ON DELETE CASCADE,  
    CONSTRAINT consult_c_discount CHECK (discount = 0 or discount = 25 or discount = 50 or  
discount = 75)  
);
```

--Crearea tabelului CONSULT\_PROCEDURA:

```
CREATE TABLE CONSULT_PROCEDURA(  
    id_consult NUMBER(4),  
    id_procedura NUMBER(4),  
    CONSTRAINT consult_fk_cp FOREIGN KEY (id_consult) REFERENCES CONSULT(id_consult)  
ON DELETE CASCADE,  
    CONSTRAINT procedura_fk_cp FOREIGN KEY (id_procedura) REFERENCES  
PROCEDURA(id_procedura) ON DELETE CASCADE,  
    CONSTRAINT consult_procedura_pk PRIMARY KEY (id_consult, id_procedura)  
);
```

Table JUDET created.

Table DOCTOR created.

Table ORAS created.

Table DIAGNOSTIC created.

Table ADRESA created.

Table RETETA created.

Table PACIENT created.

Table RETETA\_MEDICAMENT created.

Table SPITAL created.

Table PROCEDURA created.

Table MEDICAMENT created.

Table CONSULT created.

Table SPECIALIZARE created.

Table CONSULT\_PROCEDURA created.



```

-- alter pluggable database orclpdb open; din sys
alter session set container = orclpdb;
create user admin_oltp identified by admin_oltp;
grant create session to admin_oltp;
grant connect to admin_oltp;
grant create table to admin_oltp;
alter user admin_oltp quota unlimited on users;

grant select, insert, update, delete on judet to admin_oltp;
grant select, insert, update, delete on oras to admin_oltp;
grant select, insert, update, delete on adresa to admin_oltp;
grant select, insert, update, delete on pacient to admin_oltp;
grant select, insert, update, delete on spital to admin_oltp;
grant select, insert, update, delete on medicament to admin_oltp;
grant select, insert, update, delete on specializare to admin_oltp;
grant select, insert, update, delete on doctor to admin_oltp;
grant select, insert, update, delete on diagnostic to admin_oltp;
grant select, insert, update, delete on reteta to admin_oltp;
grant select, insert, update, delete on reteta_medicament to admin_oltp;
grant select, insert, update, delete on procedura to admin_oltp;
grant select, insert, update, delete on consult to admin_oltp;
grant select, insert, update, delete on consult_procedura to admin_oltp;

```

```
Session altered.
```

```
User ADMIN_OLTP created.
```

```
Grant succeeded.
```

## 2. Generarea datelor și inserarea acestora în tabele

```

var judete = ["Arad", "Bacau", "Bucuresti", "Braila", "Cluj", "Galati", "Iasi", "Suceava", "Dambovita", "Gorj"];
var orase = ["Arad", "Bacau", "Bucuresti", "Braila", "Cluj-Napoca", "Galati", "Iasi", "Suceava", "Targoviste", "Targu Jiu"];
var strazi = ["Strada Mihai Eminescu nr. 10", "Strada Rosiori nr. 217", "Strada Sibiu nr 10", "Bulevardul Unirii", "Strada Ioan Cuza nr 4"];
var coduriPostale = ["810029", "810519", "810224", "870224", "380224"];

for (i = 0; i < judete.length; i++) {
    console.log("INSERT INTO JUDET (nume_judet) VALUES (" + judete[i] + ");");
}
var b = 1;
for (i = 0; i < judete.length; i++) {
    console.log("INSERT INTO ORAS (id_judet, nume_oras) VALUES (" + b + ", " + orase[i] + ");");
    b++;
}

console.log(orase.length);

```

```

var o = 1;
for (i = 0; i < judete.length; i++) {
    for(j = 0; j < strazi.length; j++) {
        if (o == judete.length-1) {
            o = 1;
        } else {
            console.log("INSERT INTO ADRESA (id_oras, strada, cod_postal) VALUES (" + o + ", " + strazi[j] + ", " +
coduriPostale[j] + ");");
            o++;
        }
    }
};

var numePacienti = ["Popa", "Popescu", "Ion", "Ionescu", "Constantin", "Voicu", "Draghia", "Dragan", "Munteanu"];
var prenumePacienti = ["Alexandra", "Mihaela", "Oana", "Roxana", "Victoria", "Simona", "Gabriela", "Razvan",
"Maria", "Vlad", "Marius", "Ionut"];
var sex = ["F", "F", "F", "F", "F", "F", "F", "M", "F", "M", "M", "M"];
var dateNastere = ["1952-10-03", "1959-12-21", "1954-06-12", "1964-09-23", "1981-04-08", "1990-10-11",
"1988-01-14", "2019-05-25", "1958-09-11", "2016-04-03", "1998-01-04", "2018-12-11", "1943-07-13", "1952-09-09",
"1939-09-23", "1945-11-15", "2020-09-09", "2018-12-21", "1947-11-15", "2021-01-13", "2017-07-21"];

var count = 0;
var datNas = 0;
//TO_DATE('2018-10-03', 'YYYY-MM-DD')
for (i = 0; i < numePacienti.length; i++) {
    for (j = 0; j < prenumePacienti.length; j++) {
        var phonenumber = '07' + (Math.floor(Math.random() * 10) % 8 + 2) + Math.floor(Math.random() * 1e7);
        if (phonenumber.length < 10) {
            phonenumber = phonenumber + '9'.repeat(10 - phonenumber.length)
        }
        if(datNas == dateNastere.length)
        {
            datNas = 0;
        }
        if (j % 2 == 0) {
            console.log("INSERT INTO PACIENT (nume, prenume, data_nastere, telefon, email, sex) VALUES (" +
numePacienti[i] + ", " + prenumePacienti[j] + ", TO_DATE(" + dateNastere[datNas] + ", 'YYYY-MM-DD'), " +
phonenumber + ", " + prenumePacienti[j].toLocaleLowerCase() + "." + numePacienti[i].toLocaleLowerCase() +
"@gmail.com", " + sex[j] + ");");
            count++;
            datNas++;
        } else {
            console.log("INSERT INTO PACIENT (nume, prenume, data_nastere, telefon, email, sex) VALUES (" +
numePacienti[i] + ", " + prenumePacienti[j] + ", TO_DATE(" + dateNastere[datNas] + ", 'YYYY-MM-DD'), " +
phonenumber + ", " + prenumePacienti[j].toLocaleLowerCase() + "." + numePacienti[i].toLocaleLowerCase() +
"@yahoo.com", " + sex[j] + ");");
            count++;
            datNas++;
        }
    }
}
}

```

```

var index_data_nastere = 0;
var index_sex = 0;
for (i=0; i < 500; i++){
    if(index_data_nastere >= dateNastere.length){
        index_data_nastere = 0;
    }
    if(index_sex >= sex.length){
        index_sex = 0;
    }
    var phonenumber = '07' + (Math.floor(Math.random() * 10) % 8 + 2) + Math.floor(Math.random() * 1e7);
    if (phonenumber.length < 10) {
        phonenumber = phonenumber + '9'.repeat(10 - phonenumber.length)
    }
    console.log("INSERT INTO PACIENT (nume, prenume, data_nastere, telefon, email, sex) VALUES ('Nume" + i + "',
'Prenume" + i + "', TO_DATE('" + dateNastere[index_data_nastere] + "', 'YYYY-MM-DD'), '" + phonenumber + "', '" +
"prenume" + i + ".nume" + i + "@gmail.com', '" + sex[index_sex] + "');");
    count++;
    index_data_nastere++;
    index_sex++;
}

var numeSpitale = ["MedCare nr.1", "MedCare nr.2", "MedCare nr.3", "MedCare nr.4", "MedCare nr.5", "MedCare
nr.6", "MedCare nr.7", "MedCare nr.17", "MedCare nr.18", "MedCare nr.8", "MedCare nr.9", "MedCare nr.10",
"MedCare nr.11", "MedCare nr.12", "MedCare nr.13", "MedCare nr.14", "MedCare nr.15", "MedCare nr.16"];

for (i = 0; i < numeSpitale.length; i++){
    var phonenumber = '07' + (Math.floor(Math.random() * 10) % 8 + 2) + Math.floor(Math.random() * 1e7);
    if (phonenumber.length < 10) {
        phonenumber = phonenumber + '9'.repeat(10 - phonenumber.length)
    }
    console.log("INSERT INTO SPITAL (id_adresa, nume_spital, contact) VALUES(" + (i+1) + ", '" + numeSpitale[i] +
"', '" + phonenumber + "');");
}

console.log(count);
var specializari = ["oftalmologie", "alergologie", "cardiologie", "pediatrie", "pneumologie", "stomatologie"]

insert into specializare values(1, 'chirurgie', 4);
insert into specializare values(2, 'chirurgie', 5);
insert into specializare values(3, 'chirurgie', 6);

insert into specializare values(4, 'specialist', 7);
insert into specializare values(5, 'rezident', 7);
insert into specializare values(6, 'primar', 7);

insert into specializare values(7, 'chirurgie endocrina', 8);
insert into specializare values(8, 'chirurgie generala', null);

```

```
insert into specializare values(9, 'medicina interna', 12);
insert into specializare values(10, 'medicina interna', 13);
insert into specializare values(11, 'medicina interna', 14);
```

```
insert into specializare values(12, 'specialist', 15);
insert into specializare values(13, 'rezident', 15);
insert into specializare values(14, 'primar', 15);
```

```
insert into specializare values(15, 'cardiologie', 16);
insert into specializare values(16, 'gastroenterologie', null);
```

```
insert into specializare values(17, 'radiologie', 20);
insert into specializare values(18, 'radiologie', 21);
insert into specializare values(19, 'radiologie', 22);
```

```
insert into specializare values(20, 'specialist', 23);
insert into specializare values(21, 'rezident', 23);
insert into specializare values(22, 'primar', 23);
```

```
insert into specializare values(23, 'radiologie imagistica', 24);
insert into specializare values(24, 'radioterapie', null);
```

```
insert into specializare values(25, 'analize', 28);
insert into specializare values(26, 'analize', 29);
insert into specializare values(27, 'analize', 30);
```

```
insert into specializare values(28, 'specialist', 31);
insert into specializare values(29, 'rezident', 31);
insert into specializare values(30, 'primar', 31);
```

```
insert into specializare values(31, 'medicina de clinica', 32);
insert into specializare values(32, 'microbiologie medicala', null);
```

```
var medicamente = ["Paracetamol", "Panadol", "Nurofen", "Vibrocil", "Olinth", "Nospa", "Paduden", "Fervex",
"Padusin", "Emetestop", "Voltaren", "Claritine", "Diclofenat", "Baneocin", "Panthenol", "Smecta", "Augumetin",
"Zinat", "Aceclofen", "Alprazolam", "Ampicilina", "Decasept", "Ibuprofen", "Arcoxia"];
```

```
var update_medicamente = [];
```

```
for (i = 0; i < medicamente.length; i++) {
    var pret = Math.floor(Math.random() * 100) + Math.floor(Math.random() * 9)*0.1 + Math.floor(Math.random() * 5);
    console.log("INSERT INTO MEDICAMENT (nume_medicament, pret) VALUES('" + medicamente[i] + "', " + pret
+ ");");
}
```

```
var specializari_doctor = ["ORL", "Cardiologie", "Pediatrie", "Oftalmologie", "Neurologie", "Medicina Muncii",
"Urologie", "Alergologie", "Dermatologie", "Medicina Interna"];
```

```
var nume_doctori = ["Enache Oana", "Nastase Daria", "Dragomir Valentina", "Constantin Adrian", "Popa Simona
Camelia", "Voicu Luminita", "Oprea Alexandra", "Popescu Narcisa", "Calin Diana Ionela", "Florea Ana Maria", "Lascu
```

```
Dan", "Ivanov Nicoleta", "Dobre Alexandra", "Costea Marius", "Ionescu Pavel", "Scutaru Camelia", "Savu Ramona",  
"Goicea Gabriel", "Popescu Andrei", "Marin Dragos", "Cernat Ioane", "Peiu Miruna", "Tatomir Elena", "Constantinescu  
Dragomir", "Dumitrache Alexandra", "Alexandru Ionela", "Bacanu Manuel", "Matei Ana Maria", "Pana Daniela", "Popa  
Claudia", "Grecu Cristiana", "Cazacu Ramona", "Dumitrecu Andra", "Constantin Maria", "Mateiu Bogdan", "Miron  
Ionela", "Alexandrescu Mara", "Diaconescu Mariana", "Neacsu Lavinia", "Dobre Andrei"];
```

```
var dateAngajare = ["2005-10-03", "2006-12-21", "2012-06-12", "2021-09-23", "2020-04-08", "1999-10-11",  
"2016-01-14", "2018-05-25", "2011-09-11", "2017-04-03", "2020-01-04", "2013-12-11", "2009-07-13", "2014-09-09",  
"2002-09-23", "2019-11-15", "2016-09-09", "2011-12-21", "2010-11-15"];
```

```
// TO_DATE('' + dateNastere[j] + '', 'YYYY-MM-DD')
```

```
//10
```

```
console.log(specializari_doctor.length);
```

```
//19
```

```
console.log(dateAngajare.length)
```

```
//18
```

```
console.log(umeSpitale.length)
```

```
for(i=0;i<ume_doctori.length;i++)
```

```
{
```

```
    var id_spital_for_doctor = Math.floor(Math.random() * 17) + 1;
```

```
    var id_specializare_for_doctor = Math.floor(Math.random() * 40) + 1;
```

```
    var id_data_angajare_for_doctor = Math.floor(Math.random() * 18) + 1;
```

```
    console.log("INSERT INTO DOCTOR (id_spital, id_specializare, nume, data_angajare) VALUES (' +  
id_spital_for_doctor + ", " + id_specializare_for_doctor + ", " + nume_doctori[i] + ", " + "TO_DATE('" +  
dateAngajare[id_data_angajare_for_doctor] + ", 'YYYY-MM-DD'))");
```

```
}
```

```
var diagnostic = ["bronsita", "faringita", "laringita", "cancer", "hepatita", "fractura", "boli infectioase", "micoza",  
"oreion", "leucemie", "anemie", "deficit imunitar", "carenta de zinc", "carenta de magneziu", "diabet", "fibroza chistica",  
"hidrocefalia", "encefalopatia", "conjunctivita", "otita externa", "otita medie", "gripa", "pneumonie", "rinita alergica",  
"sinuzita", "ulcer gastric", "ulcer duodenal", "insolatie", "artirita reumatoida", "cifoza", "osteoporoza", "luxatie", "leziuni  
auto-provocate", "tromboza", "boli autoimune"];
```

```
var diagnosticeN = ["varicela", "variola", "rujeola", "otalgia", "otoreea", "miocardita", "insuficienta cardiaca", "angina  
pectoral", "bronsiectazia", "piotorax", "sinuzita cronica", "polip nazal", "colecistita", "urticarie", "cifoza",  
"spondilozita"];
```

```
for(i = 0; i < diagnosticeN.length; i++) {
```

```
    console.log("INSERT INTO DIAGNOSTIC (denumire_diagnostic) VALUES ('" + diagnosticeN[i] + "');");
```

```
}
```

```
var dateReteta = ["2005-10-03", "2006-12-21", "2012-06-12", "2021-09-23", "2020-04-08", "1999-10-11",  
"2016-01-14", "2018-05-25", "2011-09-11", "2017-04-03", "2020-01-04", "2013-12-11", "2009-07-13", "2014-09-09",  
"2002-09-23", "2019-11-15", "2016-09-09", "2011-12-21", "2010-11-15", "2022-01-23", "2022-02-23", "2022-03-23",  
"2022-04-23", "2022-05-23", "2022-06-23", "2022-07-23", "2022-08-23", "2022-08-19", "2022-09-23", "2022-10-23",  
"2022-11-23", "2022-12-23", "2022-04-13", "2022-06-03", "2022-09-02", "2022-01-14", "2022-09-12", "2022-03-23",  
"2023-01-19", "2023-01-07", "2021-01-23", "2021-02-23", "2021-03-23", "2021-04-23", "2021-05-23", "2021-06-23",  
"2021-07-23", "2021-08-23", "2021-09-23", "2021-10-23", "2021-11-23", "2021-12-23", "2021-07-12", "2021-03-03",  
"2021-07-10", "2021-02-12", "2021-12-10", "2022-10-02", "2021-07-22",];
```

```
var descriere = ["Dimineata si seara de 2 ori pe zi", "De 3 ori pe zi", "2 capsule inainte de fiecare masa", "Seara inainte  
de culcare", "1 comprimat la 6 ore", "O doza la 12 ore", "Aplicati circular timp de 10 minute", "Dimineata si seara  
inainte de masa", "Dimineata si seara dupa masa", "Un comprimat la 4 ore", "O pastila pe zi timp de 12 zile", "O pastila  
pe zi timp de 15 zile", "O pastila pe zi timp de 10 zile", "Dupa fiecare masa timp de 12 zile", "Dupa fiecare masa timp de
```

10 zile", "Dupa fiecare masa timp de 15 zile", "2 capsule dimineata pe stomcul gol", "2 pastile pe zi inainte de culcare", "1 comprimat la 8 ore timp de 14 zile"];

```
for(i=0; i<dateReteta.length; i++) {  
    for (j=0;j<descriere.length;j++) {  
        console.log("INSERT INTO reteta (data_reteta, descriere) VALUES (TO_DATE('" + dateReteta[i] + "',  
'YYYY-MM-DD'), '" + descriere[j] + "');");  
    }  
}
```

```
// 1121 intrari pentru retete  
// 24 medicamente
```

```
for (i=0; i<500; i++) {  
    var id_reteta = Math.floor(Math.random() * 1120) + 1;  
    var id_medicament = Math.floor(Math.random() * 23) + 1;  
    console.log("INSERT INTO RETETA_MEDICAMENT (id_reteta, id_medicament) VALUES (" + id_reteta + ", " +  
id_medicament + ");");  
}
```

```
var nume_proceduri = ["Radiografie", "Servicii imagistice", "Proceduri la nivelul aparatului respirator", "Preluare  
analize", "Proceduri ORL", "Proceduri injectabile", "Consult"];  
informatii = ["Repetat dupa 5 zile", "Repetat dupa o saptamana", "Rezultat neconcludent", "Trebuie repetat", "Consult  
dupa tratament", null, null, null, null];
```

```
for(i=0;i<nume_proceduri.length; i++) {  
    for(j=0;j<informatii.length;j++) {  
        var pret_procedura = Math.floor(Math.random() * 900) + 50;  
        if(informatii[j] == null) {  
            console.log("INSERT INTO PROCEDURA (nume_procedura, pret_procedura, informatii) VALUES (" +  
+nume_proceduri[i] + ", " + pret_procedura + ", " + informatii[j] + ");");  
        }  
        else {  
            console.log("INSERT INTO PROCEDURA (nume_procedura, pret_procedura, informatii) VALUES (" +  
+nume_proceduri[i] + ", " + pret_procedura + ", " + informatii[j] + ");");  
        }  
    }  
}
```

```
//pacienti -> 608  
//retete -> 1121  
// diagnostice -> 51  
// doctori -> 40  
//
```

```
// var dateReteta = ["2005-10-03", "2006-12-21", "2012-06-12", "2021-09-23", "2020-04-08", "1999-10-11",  
"2016-01-14", "2018-05-25", "2011-09-11", "2017-04-03", "2020-01-04", "2013-12-11", "2009-07-13", "2014-09-09",  
"2002-09-23", "2019-11-15", "2016-09-09", "2011-12-21", "2010-11-15", "2022-01-23", "2022-02-23", "2022-03-23",  
"2022-04-23", "2022-05-23", "2022-06-23", "2022-07-23", "2022-08-23", "2022-08-19", "2022-09-23", "2022-10-23",
```

```

"2022-11-23", "2022-12-23", "2022-04-13", "2022-06-03", "2022-09-02", "2022-01-14", "2022-09-12", "2022-03-23",
"2023-01-19", "2023-01-07", "2021-01-23", "2021-02-23", "2021-03-23", "2021-04-23", "2021-05-23", "2021-06-23",
"2021-07-23", "2021-08-23", "2021-09-23", "2021-10-23", "2021-11-23", "2021-12-23", "2021-07-12", "2021-03-03",
"2021-07-10", "2021-02-12", "2021-12-10", "2022-10-02", "2021-07-22",,];
//
// var descriere = ["Dimineata si seara de 2 ori pe zi", "De 3 ori pe zi", "2 capsule inainte de fiecare masa", "Seara inainte
de culcare", "1 comprimat la 6 ore", "O doza la 12 ore", "Aplicati circular timp de 10 minute", "Dimineata si seara
inainte de masa", "Dimineata si seara dupa masa", "Un comprimat la 4 ore", "O pastila pe zi timp de 12 zile", "O pastila
pe zi timp de 15 zile", "O pastila pe zi timp de 10 zile", "Dupa fiecare masa timp de 12 zile", "Dupa fiecare masa timp de
10 zile", "Dupa fiecare masa timp de 15 zile", "2 capsule dimineata pe stomcul gol", "2 pastile pe zi inainte de culcare",
"1 comprimat la 8 ore timp de 14 zile"];
//
// for(i=0; i<dateReteta.length; i++) {
//     for (j=0;j<descriere.length;j++) {
//         //console.log("INSERT INTO reteta (data_reteta, descriere) VALUES (TO_DATE('" + dateReteta[i] + '",
'YYYY-MM-DD'), '" + descriere[j] + "')");
//     }
// }
discount = [0, 25, 50, 75, 0, 0, 0, 0, 25, 25, 25, 0];
detalii = ['Pacientul prezinta simptome majore si i s-a prescris un tratament', "Pacientului i s-a administrat un tratament
pentru 12 zile", "Pacientul trebuie sa ia reteta timp de 2 saptamani", "Pacientul prezinta simptome minore", null, null,
null, null, null, null, null, null];

var c = 0;
var d = 0;
for (k=0; k<1; k++) {

    for (i = 0; i < dateReteta.length; i++) {
        var id_pacient = Math.floor(Math.random() * 607) + 1;
        var id_diagnostic = Math.floor(Math.random() * 50) + 1;
        var id_doctor = Math.floor(Math.random() * 39) + 66;
        pret_consult = Math.floor(Math.random() * 500) + 50;
        k = i + 1;
        if (k === dateReteta.length) {
            k = 1;
        }
        if (c === discount.length) {
            c = 0;
        }
        if (d === detalii.length) {
            d = 0;
        }
        if (detalii[d] === null) {
            console.log("INSERT INTO CONSULT (id_pacient, id_reteta, id_diagnostic, id_doctor, pret, data_consult,
detalii, discount) VALUES (" + id_pacient + ", " + k + ", " + id_diagnostic + ", " + id_doctor + ", " + pret_consult + ",
TO_DATE('" + dateReteta[i] + '", 'YYYY-MM-DD'), " + detalii[d] + ", " + discount[c] + ");");
            c++;
            d++;
        } else {
            console.log("INSERT INTO CONSULT (id_pacient, id_reteta, id_diagnostic, id_doctor, pret, data_consult,

```

```

detalii, discount) VALUES (" + id_pacient + ", " + k + ", " + id_diagnostic + ", " + id_doctor + ", " + pret_consult + ",
TO_DATE("'" + dateReteta[i] + "', 'YYYY-MM-DD'), '" + detalii[d] + "', " + discount[c] + ");");
    c++;
    d++;
}
}
}

// consult 575
// procedura 63
for (i=0; i<700; i++) {
    var id_consult = Math.floor(Math.random() * 574) + 1;
    var id_procedura = Math.floor(Math.random() * 62) + 1;
    console.log("INSERT INTO CONSULT_PROCEDURA (id_consult, id_procedura) VALUES (" + id_consult + ", " +
id_procedura + ");");
}

```

### 3. Crearea bazei de date depozit și a utilizatorilor

```

ALTER TABLE costuri DROP CONSTRAINT costuri_pacienti_fK;
ALTER TABLE costuri DROP CONSTRAINT costuri_spitale_fK;
ALTER TABLE costuri DROP CONSTRAINT costuri_doctori_fK;
ALTER TABLE costuri DROP CONSTRAINT costuri_retete_fK;
ALTER TABLE costuri DROP CONSTRAINT costuri_consultatii_fK;

DROP TABLE PACIENTI;
DROP TABLE SPITALE;
DROP TABLE DOCTORI;
DROP TABLE RETETE;
DROP TABLE CONSULTATII;
DROP TABLE COSTURI;
DROP TABLE TIMP;

--Crearea tabelului PACIENTI:
CREATE TABLE PACIENTI(
    id_pacient NUMBER(4),
    nume VARCHAR2(100) NOT NULL RELY,
    prenume VARCHAR2(100) NOT NULL RELY,
    sex VARCHAR2(1) CHECK (sex ='F' OR sex = 'M') ENABLE VALIDATE,
    data_nastere DATE NOT NULL RELY,
    telefon VARCHAR2(10) NOT NULL RELY,
    email VARCHAR2(30),
    categorie_pacient VARCHAR2(30) NOT NULL RELY,
    CONSTRAINT pacienti_pk PRIMARY KEY (id_pacient) RELY,
    CONSTRAINT pacienti_phone CHECK (LENGTHB(telefon) = 10) ENABLE NOVALIDATE
);

--Crearea tabelului SPITALE:
CREATE TABLE SPITALE(

```



```

id_spital NUMBER(4),
nume VARCHAR2(50) NOT NULL RELY,
contact VARCHAR2(100),
judet VARCHAR2(20),
oras VARCHAR2(20) NOT NULL RELY,
strada VARCHAR2(30),
cod_postal VARCHAR2(6),
CONSTRAINT spitale_pk PRIMARY KEY (id_spital) RELY
);

```

```

--Crearea tabelului DOCTORI
CREATE TABLE DOCTORI(
id_doctor NUMBER(5),
nume VARCHAR2(50) NOT NULL RELY,
specializare VARCHAR2(100) NOT NULL RELY,
categorie1 VARCHAR2(20),
categorie2 VARCHAR2(20),
categorie3 VARCHAR2(20),
CONSTRAINT doctori_pk PRIMARY KEY (id_doctor) RELY
);

```

```

--Crearea tabelului RETETE:
CREATE TABLE RETETE(
id_reteta NUMBER(5),
descriere VARCHAR2(200),
lista_medicamente VARCHAR2(300),
pret_reteta NUMBER(5) DEFAULT 0,
CONSTRAINT retete_pk PRIMARY KEY (id_reteta) RELY
);

```

```

--Crearea tabelului CONSULTATII:
CREATE TABLE CONSULTATII(
id_consult NUMBER(4),
denumire_diagnostic VARCHAR2(100),
lista_proceduri VARCHAR2(200),
detalii VARCHAR2(200),
pret_consult NUMBER(5) NOT NULL RELY,
discount NUMBER(2) DEFAULT 0,
CONSTRAINT consultatii_pk PRIMARY KEY (id_consult) RELY,
CONSTRAINT discount_value CHECK (discount = 0 or discount = 25 or discount = 50 or discount = 75)
RELY
);

```

```

--Crearea tabelii de TIMP
CREATE TABLE TIMP(
ID_TIMP DATE,
AN NUMBER(4,0),
SEMESTRU VARCHAR2(10 BYTE),
SEMESTRU_NR NUMBER(2),
TRIMESTRU VARCHAR2(10 BYTE),
TRIMESTRU_NR NUMBER(2),

```

```

        LUNA VARCHAR2(10 BYTE),
LUNA_NR NUMBER(2),
        SAPTAMANA_AN VARCHAR2(10 BYTE),
SAPTAMANA_AN_NR NUMBER(2),
        SAPTAMANA_LUNA VARCHAR2(10 BYTE),
SAPTAMANA_luna_NR NUMBER(2),
        ZI_AN VARCHAR2(10 BYTE),
ZI_AN_NR NUMBER(3),
        ZI_LUNA VARCHAR2(10 BYTE),
ZI_LUNA_NR NUMBER(2),
        ZI_SAPTAMANA NUMBER(4,0),
ZI_NUME VARCHAR2(30 BYTE),
        LUNA_NUME VARCHAR2(30 BYTE),
WEEKEND NUMBER(4,0),
        AN_BISECT NUMBER(4,0),
        DATA DATE,
CONSTRAINT PK_TIMP PRIMARY KEY (ID_TIMP) ENABLE VALIDATE
);

```

```

CREATE TABLE COSTURI(
    id NUMBER(4) GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by 1),
    timp_id DATE,
    doctor_id NUMBER(4),
    spital_id NUMBER(4),
    consult_id NUMBER(4),
    reteta_id NUMBER(4),
    pacient_id NUMBER(4),
    cost_consult NUMBER(5),
    cost_tratament NUMBER(5),
    CONSTRAINT costuri_pk PRIMARY KEY (id) RELY ENABLE VALIDATE,
    CONSTRAINT costuri_timp_fk FOREIGN KEY (timp_id) REFERENCES timp(id_timp) DISABLE
NOVALIDATE,
    CONSTRAINT costuri_doctori_fk FOREIGN KEY (doctor_id) REFERENCES doctori(id_doctor) ENABLE
NOVALIDATE,
    CONSTRAINT costuri_spitale_fk FOREIGN KEY (spital_id) REFERENCES spitale(id_spital) ENABLE
NOVALIDATE,
    CONSTRAINT costuri_consultatii_fk FOREIGN KEY (consult_id) REFERENCES consultatii(id_consult)
ENABLE NOVALIDATE,
    CONSTRAINT costuri_retete_fk FOREIGN KEY (reteta_id) REFERENCES retete(id_reteta) ENABLE
NOVALIDATE,
    CONSTRAINT costuri_pacienti_fk FOREIGN KEY (pacient_id) REFERENCES pacienti(id_pacient)
ENABLE NOVALIDATE
);

```

```
Table PACIENTI created.
```

```
Table SPITALE created.
```

```
Table DOCTORI created.
```

```
Table RETETE created.
```

```
Table CONSULTATII created.
```

```
Table TIMP created.
```

```
Table COSTURI created.
```

```
-- alter pluggable database orclpdb open; din sys
alter session set container = orclpdb;
create user admin_dw identified by admin_dw;
grant create session to admin_dw;
grant connect to admin_dw;
grant create table to admin_dw;
alter user admin_dw quota unlimited on users;

grant select, insert, update, delete on pacienti to admin_dw;
grant select, insert, update, delete on spitale to admin_dw;
grant select, insert, update, delete on doctori to admin_dw;
grant select, insert, update, delete on retete to admin_dw;
grant select, insert, update, delete on consultatii to admin_dw;
grant select, insert, update, delete on timp to admin_dw;
grant select, insert, update, delete on costuri to admin_dw;
```

```
Session altered.
```

```
User ADMIN_DW created.
```

```
Grant succeeded.
```

#### ***4. Popularea cu informații a bazei de date depozit folosind ca sursă datele din baza de date OLTP***

```
set serveroutput on;
```

```
create or replace function f_medicamente_list (id_ret IN NUMBER) return VARCHAR2 is
rez VARCHAR2(500);
TYPE tab_med IS TABLE OF VARCHAR2(100);
```

```

    lista_med tab_med;
begin
    select nume_medicament
    bulk collect into lista_med
    from medicament m, reteta_medicament rm
    where id_reteta = id_ret and m.id_medicament = rm.id_medicament;

    for i in lista_med.first..lista_med.last loop
        if rez is null then rez := lista_med(i);
        else rez := rez || ', ' || lista_med(i);
        end if;
    end loop;

    return rez;

exception
    when no_data_found then
        return "";
    when others then
        return "";
end;
/

create or replace function f_med_reteta_pret(id_ret IN NUMBER) return NUMBER is
    rez NUMBER(6,2) := 0;
    TYPE tab_med IS TABLE OF NUMBER(4,2);
    pret_med tab_med;
begin
    select pret
    bulk collect into pret_med
    from medicament m, reteta_medicament rm
    where id_reteta = id_ret and m.id_medicament = rm.id_medicament;

    for i in pret_med.first..pret_med.last loop
        rez := rez + pret_med(i);
    end loop;

    return rez;

exception
    when no_data_found then
        return 0;
    when others then
        return 0;
end;
/

```

```

create or replace function f_procedura_list (id_cons IN NUMBER) return VARCHAR2 is
    rez VARCHAR2(500);
    TYPE tab_proc IS TABLE OF VARCHAR2(100);
    lista_proc tab_proc;
begin
    select nume_procedura
    bulk collect into lista_proc
    from procedura p, consult_procedura cp
    where id_consult = id_cons and p.id_procedura = cp.id_procedura;

    for i in lista_proc.first..lista_proc.last loop
        if rez is null then rez := lista_proc(i);
        else rez := rez || ', ' || lista_proc(i);
        end if;
    end loop;

    return rez;

exception
    when no_data_found then
        return "";
    when others then
        return "";
end;
/

```

```

create or replace function f_categorie_pacient (id_pac IN NUMBER) return VARCHAR2 is
    rez VARCHAR2(20) := "";
    v_data DATE;
    gender VARCHAR2(1);
    varsta NUMBER(3);
begin
    select data_nastere, sex
    into v_data, gender
    from pacient
    where id_pacient = id_pac;

    select trunc(months_between(TRUNC(sysdate), v_data)/12)
    into varsta
    from dual;

    if (varsta < 18) then
        rez := 'copil';
    elsif (varsta < 61) and (gender = 'F') then
        rez := 'adult';
    elsif (varsta < 65) and (gender = 'M') then
        rez := 'adult';
    end if;
end;

```

```

else rez := 'pensionar';
end if;

return rez;

exception
  when no_data_found then
    return "";
  when others then
    return "";
end;
/

create or replace function f_categorie_doctor (id_spec IN NUMBER, nr_ierarhie IN NUMBER) return
VARCHAR2 is
  rez VARCHAR2(20) := "";
begin
  select nume_specializare
  into rez
  from specializare
  start with id_specializare = id_spec
  connect by id_specializare = prior categorie
  OFFSET nr_ierarhie ROWS FETCH NEXT 1 ROWS ONLY;

  if nr_ierarhie = 0 then return nvl(rez, 'primar');
  else return rez;
end if;

exception
  when no_data_found then
    if nr_ierarhie = 0 then return 'primar';
    else return "";
  end if;
  when others then
    if nr_ierarhie = 0 then return 'primar';
    else return "";
  end if;
end;
/

select * from specializare;

create or replace function f_consult_pret (id_cons IN NUMBER) return NUMBER is
  rez NUMBER(5) := 0;
  pret_cons NUMBER(5);
  discount_cons NUMBER(2);
  TYPE tab_proc IS TABLE OF NUMBER(5);

```

```

    pret_proc tab_proc;
begin
    select pret_procedura
    bulk collect into pret_proc
    from procedura p, consult_procedura cp
    where id_consult = id_cons and p.id_procedura = cp.id_procedura;

    for i in pret_proc.first..pret_proc.last loop
        rez := rez + pret_proc(i);
    end loop;

    select pret, discount
    into pret_cons, discount_cons
    from consult
    where id_consult = id_cons;

    rez := rez + pret_cons;
    rez := rez - discount_cons/100 * rez;

    return rez;

exception
    when no_data_found then
        return 0;
    when others then
        return 0;
end;
/

```

CREATE OR REPLACE PROCEDURE ADUCERE\_DATE IS

```

    last_time DATE;
BEGIN
    --Inserare in tabela PACIENTI
    MERGE INTO pacienti p_dw USING pacient p ON (p_dw.id_pacient = p.id_pacient)
    WHEN MATCHED THEN
        UPDATE SET
            nume = p.nume, prenume = p.prenume,
            sex = p.sex,
            telefon = p.telefon,
            email = p.email,
            categorie_pacient = f_categorie_pacient(p.id_pacient)
    WHEN NOT MATCHED THEN
        INSERT (id_pacient, nume, prenume, sex, data_nastere, telefon, email, categorie_pacient)
        VALUES (p.id_pacient, p.nume, p.prenume, p.sex, p.data_nastere, p.telefon, p.email,
        f_categorie_pacient(p.id_pacient));
    DBMS_OUTPUT.PUT_LINE('Nr. PACIENTI: ' || sql%Rowcount);

```

```

--Inserare in tabela SPITALE
MERGE INTO spitale s_dw USING (
    select id_spital, nume_spital, contact, nume_judet, nume_oras, strada, cod_postal
    from spitale s, adresa a, oras o, judet j
    where s.id_adresa = a.id_adresa and a.id_oras = o.id_oras and o.id_judet = j.id_judet
) s ON (s_dw.id_spital = s.id_spital)
WHEN MATCHED THEN
    UPDATE SET
        nume = s.nume_spital,
        contact = s.contact,
        judet = s.nume_judet,
        oras = s.nume_oras,
        strada = s.strada,
        cod_postal = s.cod_postal
WHEN NOT MATCHED THEN
    INSERT (id_spital, nume, contact, judet, oras, strada, cod_postal)
    VALUES (s.id_spital, s.nume_spital, s.contact, s.nume_judet, s.nume_oras, s.strada, s.cod_postal);
DBMS_OUTPUT.PUT_LINE('Nr. SPITALE: ' || sql%Rowcount);

--Inserare in tabela DOCTORI
MERGE INTO doctori d_dw USING (
    select id_doctor, nume, f_categorie_doctor(id_specializare, 0) AS specializare,
    f_categorie_doctor(id_specializare, 1) AS categorie1, f_categorie_doctor(id_specializare, 2) AS
    categorie2, f_categorie_doctor(id_specializare, 3) AS categorie3
    from doctori d
) d ON (d_dw.id_doctor = d.id_doctor)
WHEN MATCHED THEN
    UPDATE SET
        nume = d.nume,
        specializare = d.specializare,
        categorie1 = d.categorie1,
        categorie2 = d.categorie2,
        categorie3 = d.categorie3
WHEN NOT MATCHED THEN
    INSERT (id_doctor, nume, specializare, categorie1, categorie2, categorie3)
    VALUES (d.id_doctor, d.nume, d.specializare, d.categorie1, d.categorie2, d.categorie3);
DBMS_OUTPUT.PUT_LINE('Nr. DOCTORI: ' || sql%Rowcount);

--Insert in tabela CONSULTATII
MERGE INTO consultatii c_dw USING (
    select id_consult, denumire_diagnostic, f_procedura_list(id_consult) AS lista_proceduri, detalii, pret
    AS pret_consult, discount
    from consultatii c, diagnostic d
    where c.id_diagnostic = d.id_diagnostic
) c ON (c_dw.id_consult = c.id_consult)
-- datele vechi nu ar trebui modificate
WHEN NOT MATCHED THEN

```



```

INSERT (id_consult, denumire_diagnostic, lista_proceduri, detalii, pret_consult, discount)
VALUES (c.id_consult, c.denumire_diagnostic, c.lista_proceduri, c.detalii, c.pret_consult,
c.discount);
DBMS_OUTPUT.PUT_LINE('Nr. CONSULTATII: ' || sql%Rowcount);

--Insert in tabela RETETE
MERGE INTO retete r_dw USING (
    select id_reteta, descriere, f_medicamente_list(id_reteta) AS lista_medicamente,
f_med_reteta_pret(id_reteta) AS pret_reteta
    from reteta
) r ON (r_dw.id_reteta = r.id_reteta)
-- datele vechi nu ar trebui modificate
WHEN NOT MATCHED THEN
    INSERT (id_reteta, descriere, lista_medicamente, pret_reteta)
    VALUES (r.id_reteta, r.descriere, r.lista_medicamente, r.pret_reteta);
DBMS_OUTPUT.PUT_LINE('Nr. RETETE: ' || sql%Rowcount);

SELECT TRUNC(max(id_timp)) into last_time FROM TIMP;
MERGE INTO TIMP t_dw USING (
    SELECT
        time AS id_timp,
        TO_CHAR(time, 'yyyy') AS an,
        CASE WHEN TO_CHAR(time, 'MM') <= 6 THEN '1' ELSE '2' END AS semestru,
        CASE WHEN TO_CHAR(time, 'MM') <= 6 THEN 1 ELSE 2 END AS semestru_nr,
        CASE WHEN TO_CHAR(time, 'MM') <= 4 THEN '1' WHEN TO_CHAR(time, 'MM') >=9
THEN '3' ELSE '2' END AS trimestru,
        CASE WHEN TO_CHAR(time, 'MM') <= 4 THEN '1' WHEN TO_CHAR(time, 'MM') >=9
THEN '3' ELSE '2' END AS trimestru_nr,
        TO_CHAR(time, 'MONTH') AS luna,
        TO_CHAR(time, 'MM') AS luna_nr,
        TO_CHAR(time, 'WW') AS saptamana_an,
        TO_CHAR(time, 'WW') AS saptamana_an_nr,
        TO_CHAR(time, 'W') AS saptamana_luna,
        TO_CHAR(time, 'W') AS saptamana_luna_nr,
        TO_CHAR(time, 'DDD') AS zi_an,
        TO_CHAR(time, 'DDD') AS zi_an_nr,
        TO_CHAR(time, 'DD') AS zi_luna,
        TO_CHAR(time, 'DD') AS zi_luna_nr,
        TO_CHAR(time, 'D') AS zi_saptamana,
        TO_CHAR(time, 'DAY') AS zi_nume,
        CASE WHEN TO_CHAR(time, 'D') = 1 OR TO_CHAR(time, 'D') = 7 THEN '1' ELSE '0' END
AS weekend,
        CASE WHEN TO_CHAR(TO_DATE(TO_CHAR(time, 'yyyy') || '/03/01', 'yyyy/mm/dd'), 'DDD')
= '061' THEN 1 ELSE 0 END AS an_bisect,
        time AS data
    FROM (
        SELECT DISTINCT TRUNC(data_consult) AS time

```

```

        FROM consult
        WHERE last_time IS NULL OR data_consult > last_time
        ORDER BY time
    )
) t ON (t_dw.id_timp = t.id_timp)
-- datele vechi nu ar trebui modificate
WHEN NOT MATCHED THEN
    INSERT (id_timp, an, semestru, semestru_nr, trimestru, trimestru_nr, luna, luna_nr, saptamana_an,
saptamana_an_nr, saptamana_luna, saptamana_luna_nr, zi_an, zi_an_nr, zi_luna, zi_luna_nr,
zi_saptamana, zi_nume, weekend, an_bisect, data)
    VALUES (t.id_timp, t.an, t.semestru, t.semestru_nr, t.trimestru, t.trimestru_nr, t.luna, t.luna_nr,
t.saptamana_an, t.saptamana_an_nr, t.saptamana_luna, t.saptamana_luna_nr, t.zi_an, t.zi_an_nr, t.zi_luna,
t.zi_luna_nr, t.zi_saptamana, t.zi_nume, t.weekend, t.an_bisect, t.data);
    DBMS_OUTPUT.PUT_LINE('Nr. Timpi: ' || sql%Rowcount);

--Insert in tabel COSTURI
MERGE INTO costuri c_dw USING (
    select c.data_consult AS timp_id, c.id_doctor AS doctor_id, d.id_spital AS spital_id, c.id_consult
AS consult_id, c.id_reteta AS reteta_id, c.id_pacient AS pacient_id, f_consult_pret(c.id_consult) AS
cost_consult, f_med_reteta_pret(c.id_reteta) AS cost_tratament
    from consult c, doctor d
    where c.id_doctor = d.id_doctor
) c ON (c_dw.timp_id = c.timp_id AND c_dw.doctor_id = c.doctor_id AND c_dw.spital_id =
c.spital_id AND c_dw.consult_id = c.consult_id AND c_dw.reteta_id = c.reteta_id AND c_dw.pacient_id
= c.pacient_id)
-- datele vechi nu ar trebui modificate
WHEN NOT MATCHED THEN
    INSERT (timp_id, doctor_id, spital_id, consult_id, reteta_id, pacient_id, cost_consult,
cost_tratament)
    VALUES (c.timp_id, c.doctor_id, c.spital_id, c.consult_id, c.reteta_id, c.pacient_id, c.cost_consult,
c.cost_tratament);
    DBMS_OUTPUT.PUT_LINE('Nr. COSTURI: ' || sql%Rowcount);
END;
/

EXEC ADUCERE_DATE();

```

Function F_MEDICAMENTE_LIST compiled	
Function F_MED_RETETA_PRET compiled	Procedure ADUCERE_DATE compiled
Function F_PROCEDURA_LIST compiled	Nr. PACIENTI: 608
Function F_CATEGORIE_PACIENT compiled	Nr. SPITALE: 18
Function F_CATEGORIE_DOCTOR compiled	Nr. DOCTORI: 31
Function F_CONSULT_PRET compiled	Nr. CONSULTATII: 109
	Nr. RETETE: 1120
	Nr. Timpi: 55
	Nr. COSTURI: 109
Procedure ADUCERE_DATE compiled	PL/SQL procedure successfully completed.

## 5. Definirea constrângerilor

Constrângerile au fost definite odată cu crearea tabelelor Data Warehouse-ului și au fost explicate în raportul de analiză.

Am realizat un select pentru a vedea cum arată constrângerile pentru fiecare tabel.

```
select constraint_name, constraint_type, table_name, search_condition, status, validated,
rely
from user_constraints
where table_name = <tab_name>;
```

### Constrângerile tabelului PACIENTI:

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION	STATUS	VALIDATED	RELY
1 SYS_C009026	C	PACIENTI	"NUME" IS NOT NULL	ENABLED	VALIDATED	RELY
2 SYS_C009027	C	PACIENTI	"PRENUME" IS NOT NULL	ENABLED	VALIDATED	RELY
3 SYS_C009028	C	PACIENTI	"DATA_NASTERE" IS NOT NULL	ENABLED	VALIDATED	RELY
4 SYS_C009029	C	PACIENTI	"TELEFON" IS NOT NULL	ENABLED	VALIDATED	RELY
5 SYS_C009030	C	PACIENTI	"CATEGORIE_PACIENT" IS NOT NULL	ENABLED	VALIDATED	RELY
6 SYS_C009031	C	PACIENTI	sex = 'F' OR sex = 'M'	ENABLED	VALIDATED	(null)
7 PACIENTI_PHONE	C	PACIENTI	LENGTHB(telefon) = 10	ENABLED	NOT VALIDATED	(null)
8 PACIENTI_PK	P	PACIENTI	(null)	ENABLED	VALIDATED	RELY

### Constrângerile tabelului SPITALE:

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION	STATUS	VALIDATED	RELY
1 SYS_C009114	C	SPITALE	"NUME" IS NOT NULL	ENABLED	VALIDATED	RELY
2 SYS_C009115	C	SPITALE	"ORAS" IS NOT NULL	ENABLED	VALIDATED	RELY
3 SPITALE_PK	P	SPITALE	(null)	ENABLED	VALIDATED	RELY

### Constrângerile tabelului DOCTORI:

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION	STATUS	VALIDATED	RELY
1 SYS_C009037	C	DOCTORI	"NUME" IS NOT NULL	ENABLED	VALIDATED	RELY
2 SYS_C009038	C	DOCTORI	"SPECIALIZARE" IS NOT NULL	ENABLED	VALIDATED	RELY
3 DOCTORI_PK	P	DOCTORI	(null)	ENABLED	VALIDATED	RELY

### Constrângerile tabelului CONSULTAȚII:

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION	STATUS	VALIDATED	RELY
1 SYS_C009041	C	CONSULTATII	"PRET_CONSULT" IS NOT NULL	ENABLED	VALIDATED	RELY
2 DISCOUNT_VALUE	C	CONSULTATII	discount = 0 or discount = 25 or discount = 50 or discount = 75	ENABLED	VALIDATED	RELY
3 CONSULTATII_PK	P	CONSULTATII	(null)	ENABLED	VALIDATED	RELY

### Constrângerile tabelului REȚETE:

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION	STATUS	VALIDATED	RELY
1 RETETE_PK	P	RETETE	(null)	ENABLED	VALIDATED	RELY

### Constrângerile tabelului TIMP:

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION	STATUS	VALIDATED	RELY
1 PK_TIMP	P	TIMP	(null)	ENABLED	VALIDATED	(null)

### Constrângerile tabelului COSTURI:

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION	STATUS	VALIDATED	RELY
1 COSTURI_DOCTORI_FK	R	COSTURI	(null)	ENABLED	NOT VALIDATED	(null)
2 COSTURI_SPITALE_FK	R	COSTURI	(null)	ENABLED	NOT VALIDATED	(null)
3 COSTURI_CONSULTATII_FK	R	COSTURI	(null)	ENABLED	NOT VALIDATED	(null)
4 COSTURI_RETETE_FK	R	COSTURI	(null)	ENABLED	NOT VALIDATED	(null)
5 COSTURI_PACIENTI_FK	R	COSTURI	(null)	ENABLED	NOT VALIDATED	(null)
6 COSTURI_PK	P	COSTURI	(null)	ENABLED	VALIDATED	(null)
7 COSTURI_TIMP_FK	R	COSTURI	(null)	DISABLED	NOT VALIDATED	(null)

## 6. Definirea indecșilor și a cererilor SQL însoțite de planul de execuție al acestora

```
CREATE BITMAP INDEX ind_bmp_doctori_titulatura
ON doctori(categorie2);
```

```
analyze index ind_bmp_doctori_titulatura compute statistics;
```

```
CREATE BITMAP INDEX ind_bmp_pacienti_cat
ON pacienti(categorie_pacient);
```

```
analyze index ind_bmp_pacienti_cat compute statistics;
```

```
CREATE BITMAP INDEX ind_bmp_consult_disc
ON consultatii(discount);
```

```
analyze index ind_bmp_pacienti_cat compute statistics;
```

```

INDEX IND_BMP_DOCTORI_TITULATURA created.

Index IND_BMP_DOCTORI_TITULATURA analyzed.

INDEX IND_BMP_PACIENTI_CAT created.

Index IND_BMP_PACIENTI_CAT analyzed.

INDEX IND_BMP_CONSULT_DISC created.

Index IND_BMP_CONSULT_DISC analyzed.

```

Pentru rezolvarea cererii din raportul de analiză am creat și un index pe join-ul tabelelor COSTURI și PACIENȚI pe coloana *categorie\_pacient*. În acest mod am optimizat costul cererii.

```

CREATE BITMAP INDEX ind_bmp_join_pens_f_costuri
ON costuri(categorie_pacient)
FROM costuri c, pacienti d
WHERE c.pacient_id = d.id_pacient local;

EXPLAIN PLAN
SET STATEMENT_ID = 's1_index_cerere' FOR
select
  (select count (*) from doctori where lower(categorie2) = 'rezident') nr_doctori,
  (select avg(pret_consult) from consultatii where discount > 25) pret_mediu,
  (select
    sum (cost_tratament)
  from pacienti p, costuri c, timp t
  where categorie_pacient = 'pensionar'
  and c.pacient_id = p.id_pacient
  and c.timp_id = t.id_timp
  and t.luna = 12
  and t.an = 2021) cost
from dual;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','s1_index_cerere','serial'));

```

PLAN_TABLE_OUTPUT								
Plan hash value: 976532607								
Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop
0	SELECT STATEMENT		1		3 (0)	00:00:01		
1	SORT AGGREGATE		1	12				
2	BITMAP CONVERSION TO ROWIDS		1	12				
* 3	BITMAP INDEX FAST FULL SCAN	IND_BMP_DOCTORI_TITULATURA						
4	SORT AGGREGATE		1	26				
5	TABLE ACCESS BY INDEX ROWID BATCHED	CONSULTATII	1	26	0 (0)	00:00:01		
6	BITMAP CONVERSION TO ROWIDS							
* 7	BITMAP INDEX RANGE SCAN	IND_BMP_CONSULT_DISC						
8	SORT AGGREGATE		1	64				
9	NESTED LOOPS		1	64	1 (0)	00:00:01		
10	NESTED LOOPS		1	64	1 (0)	00:00:01		
11	PARTITION RANGE ALL		1	35	1 (0)	00:00:01	1	5
12	TABLE ACCESS BY LOCAL INDEX ROWID BATCHED	COSTURI	1	35	1 (0)	00:00:01	1	5
13	BITMAP CONVERSION TO ROWIDS							
* 14	BITMAP INDEX SINGLE VALUE	IND_BMP_JOIN_PENS_F_COSTURI					1	5
* 15	INDEX UNIQUE SCAN	PK_TIMP	1		0 (0)	00:00:01		
* 16	TABLE ACCESS BY INDEX ROWID	TIMP	1	29	0 (0)	00:00:01		
17	FAST DUAL		1		2 (0)	00:00:01		

Se poate observa că sunt folosiți și indecșii pentru coloana *discount* și pentru coloana *categorie2*.

## 7. Definirea obiectelor de tip dimensiune, validarea acestora

```
CREATE DIMENSION adresa_dim
LEVEL spital IS (spitale.id_spital)
LEVEL cod_postal IS (spitale.cod_postal)
LEVEL strada IS (spitale.strada)
LEVEL oras IS (spitale.oras)
LEVEL judet is (spitale.judet)
HIERARCHY h_adresa (spital
    CHILD OF cod_postal
    CHILD OF strada
    CHILD OF oras
    CHILD OF judet)
ATTRIBUTE spital DETERMINES(spitale.numa, spitale.contact);

EXECUTE DBMS_DIMENSION.VALIDATE_DIMENSION ('adresa_dim',FALSE, TRUE, 'st_id');
```





```
Dimension ADRESA_DIM created.
```

```
PL/SQL procedure successfully completed.
```

select \* from dimension\_exceptions where statement\_id = 'st\_id';

Script Output x

Query Result x

All Rows Fetched: 0 in 0.011 seconds

STATEME...	OWNER	TABLE_N...	DIMENSI...	RELATIO...	BAD_RO...
------------	-------	------------	------------	------------	-----------

```
CREATE DIMENSION doctori_dim
LEVEL doctor IS (doctori.id_doctor)
LEVEL specializare IS (doctori.specializare)
LEVEL titulatura IS (doctori.categorie1)
LEVEL subdomeniu IS (doctori.categorie2)
```

```
LEVEL domeniu IS (doctori.categorie3)
```

```
HIERARCHY h_doctor (doctor
```

```
    CHILD OF specializare
```

```
    CHILD OF titlatura
```

```
    CHILD OF subdomeniu
```

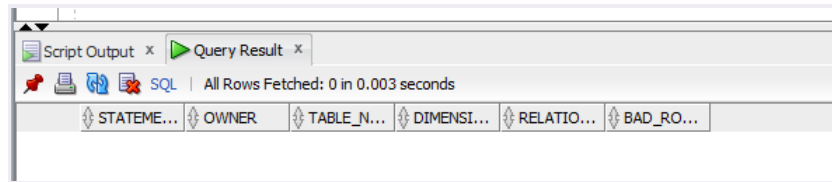
```
    CHILD OF domeniu);
```

```
EXECUTE DBMS_DIMENSION.VALIDATE_DIMENSION ('doctori_dim',FALSE, TRUE, 'st_id2');
```

```
Dimension DOCTORI_DIM created.
```

```
PL/SQL procedure successfully completed.
```

```
select * from dimension_exceptions where statement_id = 'st_id2';
```



## 8. Definirea partițiilor; definirea cererilor SQL însoțite de planul de execuție al acestora din care să reiasă ca optimizorul utilizează eficient partițiile

```
DROP TABLE COSTURI;
```

```
DROP TABLE SPITALE;
```

```
--Partitionare tabela SPITALE
```

```
CREATE TABLE SPITALE (
```

```
    id_spital NUMBER(4),
```

```
    nume VARCHAR2(50) NOT NULL RELY,
```

```
    contact VARCHAR2(100),
```

```
    judet VARCHAR2(20),
```

```
    oras VARCHAR2(20) NOT NULL RELY,
```

```
    strada VARCHAR2(30),
```

```
    cod_postal VARCHAR2(6),
```

```
    CONSTRAINT spitale_pk PRIMARY KEY (id_spital) RELY
```

```
)
```

```
PARTITION BY LIST(judet) (
```

```
    PARTITION judete_moldova VALUES('Botosani', 'Bacau', 'Galati', 'Iasi', 'Neamt', 'Suceava', 'Vaslui'),
```

```
    PARTITION judete_muntenia VALUES('Arges', 'Braila', 'Buzau', 'Calarasi', 'Dambovita', 'Giurgiu', 'Ialomita', 'Ilfov', 'Prahova', 'Teleorman', 'Municipiul Bucuresti'),
```

```
    PARTITION judete_oltenia VALUES('Dolj', 'Gorj', 'Mehedinti', 'Olt', 'Valcea'),
```

```
    PARTITION judete_banat VALUES('Timis', 'Caras-Severin', 'Arad'),
```

```
    PARTITION judete_transilvania VALUES('Alba', 'Bistrita-Nasaud', 'Brasov', 'Cluj', 'Sibiu', 'Mures', 'Hunedoara'),
```

```
    PARTITION judete_alteale VALUES(DEFAULT));
```

```
--Partitionare tabela COSTURI
```

```
CREATE TABLE COSTURI (
```

```

id NUMBER(4),
timp_id DATE,
doctor_id NUMBER(4),
spital_id NUMBER(4),
consult_id NUMBER(4),
reteta_id NUMBER(4),
pacient_id NUMBER(4),
cost_consult NUMBER(5),
cost_tratament NUMBER(5),
CONSTRAINT costuri_pk PRIMARY KEY (id) ENABLE VALIDATE,
CONSTRAINT costuri_timp_fk FOREIGN KEY (timp_id) REFERENCES timp(id_timp) DISABLE
NOVALIDATE,
CONSTRAINT costuri_doctori_fk FOREIGN KEY (doctor_id) REFERENCES doctori(id_doctor) ENABLE
NOVALIDATE,
CONSTRAINT costuri_spitale_fk FOREIGN KEY (spital_id) REFERENCES spitale(id_spital) ENABLE
NOVALIDATE,
CONSTRAINT costuri_consultatii_fk FOREIGN KEY (consult_id) REFERENCES consultatii(id_consult)
ENABLE NOVALIDATE,
CONSTRAINT costuri_retete_fk FOREIGN KEY (reteta_id) REFERENCES retete(id_reteta) ENABLE
NOVALIDATE,
CONSTRAINT costuri_pacienti_fk FOREIGN KEY (pacient_id) REFERENCES pacienti(id_pacient) ENABLE
NOVALIDATE
)
PARTITION BY RANGE(timp_id) (
PARTITION trimestrul_I VALUES LESS THAN(TO_DATE('01/04/2022','DD/MM/YYYY')),
PARTITION trimestrul_II VALUES LESS THAN(TO_DATE('01/07/2022','DD/MM/YYYY')),
PARTITION trimestrul_III VALUES LESS THAN(TO_DATE('01/10/2022','DD/MM/YYYY')),
PARTITION trimestrul_IV VALUES LESS THAN(TO_DATE('01/01/2023','DD/MM/YYYY')),
PARTITION rest VALUES LESS THAN (MAXVALUE));

```

Table COSTURI dropped.

Table SPITALE dropped.

Table SPITALE created.

Table COSTURI created.

## 9. Optimizarea cererii SQL propusă în etapa de analiză

-- Sa se determine numărul lunar de consulturi din anul cu cele mai multe consulturi (în caz de egalitate, se alege cel mai recent an)

```

EXPLAIN PLAN SET STATEMENT_ID = 'DWBI_9_a' FOR
SELECT luna_nr, count(luna_nr)
FROM consultatii
  INNER JOIN costuri ON id_consult = consult_id
  INNER JOIN timp ON timp_id = id_timp
WHERE an = (
  SELECT an

```



```

FROM consultatii
  INNER JOIN costuri ON id_consult = consult_id
  INNER JOIN timp ON timp_id = id_timp
GROUP BY an
ORDER BY COUNT(an) DESC, an DESC
FETCH FIRST 1 ROWS ONLY
)
GROUP BY luna_nr
ORDER BY luna_nr;

```

```

SELECT plan_table_output FROM table(dbms_xplan.display('plan_table', 'DWBI_9_a', 'serial'));
-----

```

```

EXPLAIN PLAN SET STATEMENT_ID = 'DWBI_9_b' FOR
WITH

```

```

  consulturi_cu_data AS (
    select an, luna_nr
    FROM consultatii
      INNER JOIN costuri ON id_consult = consult_id
      INNER JOIN timp ON timp_id = id_timp
  ),
  an_cerut AS (
    SELECT an
    FROM consulturi_cu_data
    GROUP BY an
    ORDER BY COUNT(an) DESC, an DESC
    FETCH FIRST 1 ROWS ONLY
  )
SELECT luna_nr, count(luna_nr)
FROM consulturi_cu_data
WHERE an = (SELECT an FROM an_cerut)
GROUP BY luna_nr
ORDER BY luna_nr;

```

```

SELECT plan_table_output FROM table(dbms_xplan.display('plan_table', 'DWBI_9_b', 'serial'));
-----

```

```

EXPLAIN PLAN SET STATEMENT_ID = 'DWBI_9_c' FOR
WITH

```

```

  consulturi_cu_data AS (
    SELECT an, luna_nr, COUNT(luna_nr) AS lunar_count
    FROM consultatii
      INNER JOIN costuri ON id_consult = consult_id
      INNER JOIN timp ON timp_id = id_timp
    GROUP BY an, luna_nr
  ),
  an_cerut AS (
    SELECT an
    FROM consulturi_cu_data
    GROUP BY an
    ORDER BY SUM(lunar_count) DESC, an DESC
  )

```

```

        FETCH FIRST 1 ROWS ONLY
    )
SELECT luna_nr, lunar_count
FROM consulturi_cu_data
WHERE an = (SELECT an FROM an_cerut)
ORDER BY luna_nr;

SELECT plan_table_output FROM table(dbms_xplan.display('plan_table', 'DWBI_9_c','serial'));
-----

EXPLAIN PLAN SET STATEMENT_ID = 'DWBI_9_d' FOR
WITH
consulturi_cu_data AS (
    SELECT an, luna_nr, COUNT(luna_nr) AS lunar_count
    FROM consultatii
        INNER JOIN costuri ON id_consult = consult_id
        INNER JOIN timp ON timp_id = id_timp
    GROUP BY an, luna_nr
    ORDER BY luna_nr
),
an_cerut AS (
    SELECT an
    FROM consulturi_cu_data
    GROUP BY an
    ORDER BY SUM(lunar_count) DESC, an DESC
    FETCH FIRST 1 ROWS ONLY
)
SELECT luna_nr, lunar_count
FROM consulturi_cu_data cce_extern
WHERE an = (SELECT an FROM an_cerut);

SELECT plan_table_output FROM table(dbms_xplan.display('plan_table', 'DWBI_9_d','serial'));

```

## 10. Crearea rapoartelor cu complexitate diferită

1. Afișați procentele medicamentelor vândute (raceala - Paracetamol, antiinflamator - Diclofenac, durere - Voltaren, stomac - No-spa) în anul în care s-a înregistrat cel mai mare cost al unui tratament (în caz de egalitate, se va lua cel mai recent an).

```

SELECT
    SUM(INSTR(lista_medicamente, 'Paracetamol')) AS paracetamol,
    SUM(INSTR(lista_medicamente, 'Diclofenat')) AS diclofenat,
    SUM(INSTR(lista_medicamente, 'Voltaren')) AS voltaren,
    SUM(INSTR(lista_medicamente, 'Nospa')) AS nospa
FROM retete INNER JOIN (
    -- lista retetelor care au fost prescrise in anul cerut
    SELECT reteta_id AS id_reteta
    FROM costuri INNER JOIN timp ON timp_id = id_timp
    WHERE an = (

```

```

-- primul an din rankingul anilor in functie de costul maxim al unui tratament
SELECT an
FROM costuri INNER JOIN timp ON timp_id = id_timp
GROUP BY an
ORDER BY MAX(cost_tratament) DESC, an DESC
FETCH FIRST 1 ROWS ONLY
)
) USING (id_reteta)
WHERE lista_medicamente IS NOT NULL;

```

2. Să se afișeze evoluția mediilor lunare a costurilor pentru tratamentul pacienților care au fost diagnosticați cu cancer din anul 2022 (anul reprezintă durata pe care se calculează mediile, nu perioada diagnosticării).

```

SELECT luna_nr, AVG(pret_final) AS pret_mediu
FROM (
-- selectam consulturile (id-ul) si pretul final al acestora pentru boala ceruta
SELECT id_consult, pret_consult * (1-NVL(discount,0)/100) AS pret_final
FROM consultatii
WHERE denumire_diagnostic = 'cancer'
) INNER JOIN (
-- selectam toate consulturile (id si data) efectuate in anul cerut
SELECT consult_id AS id_consult, luna_nr
FROM costuri INNER JOIN timp ON timp_id = id_timp
WHERE an = 2022
) USING (id_consult)
GROUP BY luna_nr
ORDER BY 1;

```

3. Să se obțină valoarea costului consulturilor din anul 2022 al fiecărui spital din București.

```

SELECT nume, SUM(cost_consult)
FROM costuri
INNER JOIN spitale s ON spital_id = id_spital
INNER JOIN timp ON timp_id = id_timp
WHERE an = 2022 AND oras = 'Bucuresti'
GROUP BY id_spital, nume
ORDER BY nume;

```

4. Să se obțină numele de la top 10 pacienți (separați pe categorie de sex) care au avut cele mai scumpe consulturi fără discount.

```

WITH pacienti_ordonati AS (
SELECT nume, prenume, sex

```

```

FROM consultatii
  INNER JOIN costuri ON id_consult = consult_id
  INNER JOIN pacienti ON pacient_id = id_pacient
  INNER JOIN timp ON timp_id = id_timp
WHERE NVL(discount, 0) = 0 AND an = 2021
ORDER BY pret_consult DESC
)
(SELECT *
  FROM pacienti_ordonati
  WHERE sex = 'F'
  FETCH FIRST 10 ROWS ONLY)
UNION
(SELECT *
  FROM pacienti_ordonati
  WHERE sex = 'M'
  FETCH FIRST 10 ROWS ONLY);

```

5. Să se determine numerele din fiecare categorie de vârstă al pacienților care și-au făcut analizele la spitalele din București (se ia în considerare vârsta la momentul recoltării).

```

SELECT categorie_pacient, count(categorie_pacient) AS total
FROM (
  SELECT CASE
    WHEN age < 18 THEN 'copil'
    WHEN age < 61 AND sex = 'F' THEN 'adult'
    WHEN age < 65 AND sex = 'M' THEN 'adult'
    ELSE 'pensionar' END AS categorie_pacient
  FROM (
    SELECT TRUNC(MONTHS_BETWEEN(data,data_nastere)/12) AS age, sex
    FROM consultatii
      INNER JOIN costuri ON id_consult = consult_id
      INNER JOIN spitale ON id_spital = spital_id
      INNER JOIN pacienti ON id_pacient = pacient_id
      INNER JOIN timp ON id_timp = timp_id
    WHERE lista_proceduri LIKE '%Preluare analize%'
      AND oras = 'Bucuresti'
    )
  )
GROUP BY categorie_pacient;

```

*IMPLEMENTAREA APLICATIEI*  
**Data Warehouse & Business Intelligence**

**Echipa 4:**

*Dobre Mihaela Beatrice*

*Mocanu Alina Cristina*

*Muşat Andreea*

*Surcea Mihai Daniel*

## ❖ Adauga un doctor in sistem

### Adauga un doctor

Nume

Data angajarii



Spital

Specializare

## ❖ Adauga un pacient nou

### Adauga un pacient

Nume

Prenume

Data nasterii




Telefon

Email

Introduceti sexul pacientului




❖ Lista doctorilor



Enache Oana

STERGE DOCTOR


EDITEAZA DOCTOR



Nastase Daria

STERGE DOCTOR


EDITEAZA DOCTOR



Dragomir Valentina

STERGE DOCTOR

EDITEAZA DOCTOR



Constantin Adrian

STERGE DOCTOR

EDITEAZA DOCTOR

< previous

1

2

3

4

...

8

9

10

next >

❖ Tabelul tuturor pacientilor

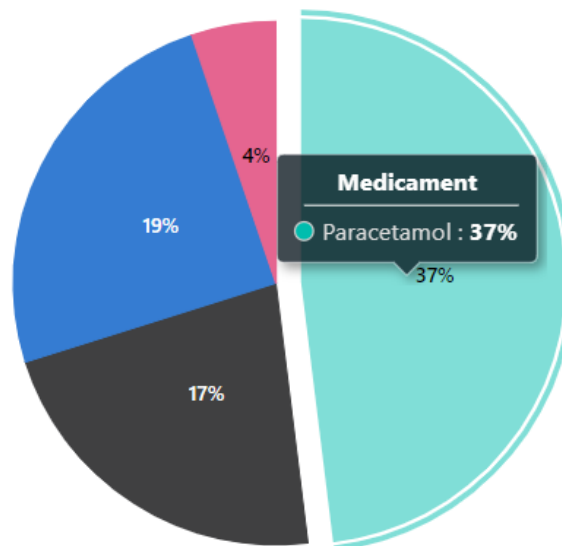
Lista pacientilor

#	Nume	Prenume	Telefon	Email	Setari	
391	Nume282	Prenume282	2016-04-02T22:00:00.000Z	0749197107	Sterge pacientul	Editeaza
392	Nume283	Prenume283	1998-01-03T22:00:00.000Z	0772937550	Sterge pacientul	Editeaza
393	Nume284	Prenume284	2018-12-10T22:00:00.000Z	0762510550	Sterge pacientul	Editeaza
394	Nume285	Prenume285	1943-07-12T22:00:00.000Z	0779893077	Sterge pacientul	Editeaza
395	Nume286	Prenume286	1952-09-08T22:00:00.000Z	0736259307	Sterge pacientul	Editeaza
396	Nume287	Prenume287	1939-09-22T22:00:00.000Z	0783472024	Sterge pacientul	Editeaza
397	Nume288	Prenume288	1945-11-14T22:00:00.000Z	0783196129	Sterge pacientul	Editeaza
398	Nume289	Prenume289	2020-09-08T22:00:00.000Z	0798752369	Sterge pacientul	Editeaza
399	Nume290	Prenume290	2018-12-20T22:00:00.000Z	0742207329	Sterge pacientul	Editeaza
400	Nume291	Prenume291	1947-11-14T22:00:00.000Z	0728620667	Sterge pacientul	Editeaza

## ❖ Rapoartele grafice

1. Procentele medicamentelor vandute (pentru Paracetamol, Diclofenac, Voltare, No-spa).

Procentele medicamentelor vândute



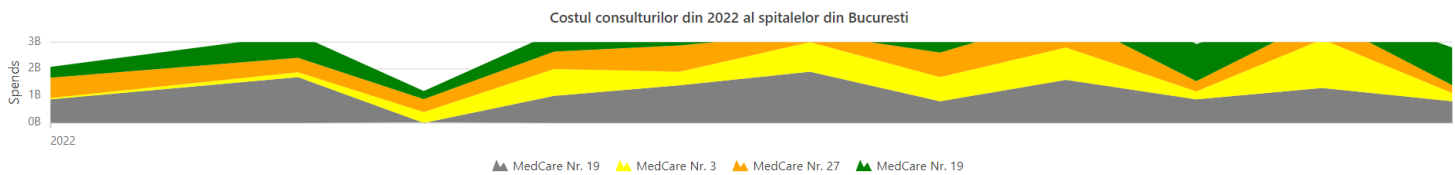
2. Evolutia costurilor din anul 2022 pentru tratamentul pacientilor diagnosticati cu cancer.

Evolutia costurilor pentru tratamentul de cancer - anul 2022



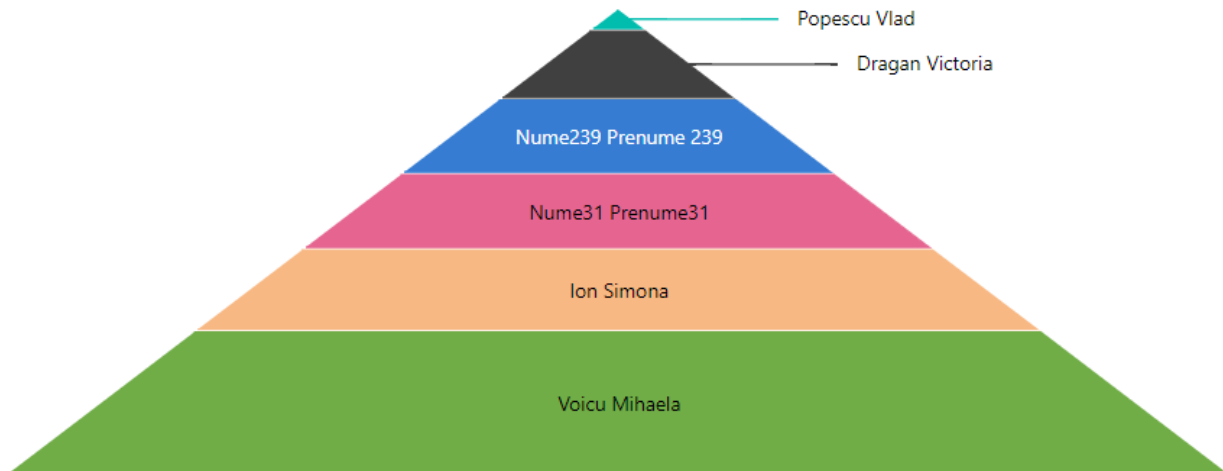


### 3. Costul tuturor consulturilor din anul 2022 pentru fiecare spital din Bucuresti.



### 4. Topul pacientilor care au avut cele mai scumpe consulturi (fara un discount aplicat).

#### Top 6 pacienți (cele mai scumpe consulturi fără discount)



5. Procentele analizelor in functie de categoria pacientilor (copii, adulti, pensionari).

