**Electron Hospital Management System**

Contents

[1. Problem domain description 1](#_Toc200392263)

[2. Logical model (Data Modeler) 2](#_Toc200392264)

[3. Relational model (Data Modeler) 3](#_Toc200392265)

[4. Physical model (Use for SQL developer) 3](#_Toc200392266)

[5. Data population 9](#_Toc200392267)

[6. PL/SQL (beginner to advanced) 17](#_Toc200392268)

# 1. Problem domain description

The **Electron Hospital Management System** is a database-driven web application designed to streamline hospital operations, including patient records, doctor services, appointments, and billing. This project models a real-world hospital management system with a focus on relational database design, integrity constraints, and efficient data retrieval.

The “Patients” entity has 5 attributes: PatientID as primary key, PatientName, BirthDate, Gender, Contact (the phone number of patient). Each patient has a unique medical record, and each medical record has a patient, so it’s 1:1 relationship.

Patients can view available services in hospital. The “Services” entity and “Doctors” entity are related with many-to-many relationship, because one doctor can have more than one specialties (ex: same doctor as pulmonologist and allergologist) or can offer diagnostics, consultations, even surgeries. On the other hand, 1 service can be offered by many doctors also.

Then, 1 patient can make as much as appointments (s)he wants. Also, doctors and appointments have 1: many relationships:1 doctor can have multiple appointments.

Finally, patients can view their bills. 1 patient can have multiple bills, for example each for different services.

Most of the attributes are important and must not be null. But we have optional attribute ‘prescription’ in ‘MedicalRecords’ because maybe patients even don’t need prescription for any medicine. Also, other optional attribute is ‘VisitReason’ in ‘Appointments’ for serving patients better knowing beforehand what problem specifically they have. For the primary keys, hospital uses artificially generated fixed number of characters for being unique and not changing over time.

# 2. Logical model (Data Modeler)

A screenshot of a computer

Description automatically generated

# 3. Relational model (Data Modeler)

*A screenshot of a computer

Description automatically generated*

# 4. Physical model (Use for SQL developer)

**-- You can export your relational model as DDL file, then make some changes,fix mistakes and paste into SQL developer.**

**-- Patients**

CREATE TABLE Patients (

PatientID CHAR(7) NOT NULL,

PatientName VARCHAR2(50) NOT NULL,

BirthDate DATE NOT NULL,

Gender VARCHAR2(10) NOT NULL,

Contact VARCHAR2(15) NOT NULL

);

ALTER TABLE Patients

ADD CONSTRAINT Patients\_PK PRIMARY KEY (PatientID);

**-- Doctors**

CREATE TABLE Doctors (

DoctorID CHAR(7) NOT NULL,

DoctorName VARCHAR2(50) NOT NULL,

Phone VARCHAR2(15) NOT NULL,

WorkingHours VARCHAR2(150) NOT NULL

);

ALTER TABLE Doctors

ADD CONSTRAINT Doctors\_PK PRIMARY KEY (DoctorID);

**-- Services**

-- Services

CREATE TABLE Services (

ServiceCode CHAR(6) NOT NULL,

Specialty VARCHAR2(150) NOT NULL,

Price\_Euro NUMBER(10,2) NOT NULL,

DoctorID Char(7) Not Null

);

ALTER TABLE Services

ADD CONSTRAINT Services\_PK PRIMARY KEY (ServiceCode);

ALTER TABLE Services

ADD CONSTRAINT Services\_Doctors\_FK FOREIGN KEY (DoctorID)

REFERENCES Doctors (DoctorID);

**-- Doctor\_Service**

CREATE TABLE Doctor\_Service (

Doctors\_DoctorID CHAR(7) NOT NULL,

Services\_ServiceCode CHAR(6) NOT NULL

);

ALTER TABLE Doctor\_Service

ADD CONSTRAINT Doctor\_Service\_PK PRIMARY KEY (Doctors\_DoctorID, Services\_ServiceCode);

ALTER TABLE Doctor\_Service

ADD CONSTRAINT Doctor\_Service\_Doctors\_FK FOREIGN KEY (Doctors\_DoctorID)

REFERENCES Doctors (DoctorID);

ALTER TABLE Doctor\_Service

ADD CONSTRAINT Doctor\_Service\_Services\_FK FOREIGN KEY (Services\_ServiceCode)

REFERENCES Services (ServiceCode);

**-- Appointments**

CREATE TABLE Appointments (

AppointmentID CHAR(7) NOT NULL,

Datetime DATE NOT NULL,

VisitReason VARCHAR2(150), -- Optional

PatientID CHAR(7) NOT NULL,

DoctorID CHAR(7) NOT NULL

);

ALTER TABLE Appointments

ADD CONSTRAINT Appointments\_PK PRIMARY KEY (AppointmentID);

ALTER TABLE Appointments

ADD CONSTRAINT Appointments\_Doctors\_FK FOREIGN KEY (DoctorID)

REFERENCES Doctors (DoctorID);

ALTER TABLE Appointments

ADD CONSTRAINT Appointments\_Patients\_FK FOREIGN KEY (PatientID)

REFERENCES Patients (PatientID);

**-- Bills**

CREATE TABLE Bills (

BillID CHAR(6) NOT NULL,

Amount\_Euro NUMBER(10,2) NOT NULL,

PaymentStatus VARCHAR2(15) NOT NULL,

"Date" DATE NOT NULL,

PatientID CHAR(7) NOT NULL

);

ALTER TABLE Bills

ADD CONSTRAINT Bills\_PK PRIMARY KEY (BillID);

ALTER TABLE Bills

ADD CONSTRAINT Bills\_Patients\_FK FOREIGN KEY (PatientID)

REFERENCES Patients (PatientID);

**-- MedicalRecords**

CREATE TABLE MedicalRecords (

RecordID CHAR(6) NOT NULL,

PatientID CHAR(7) NOT NULL,

Diagnosis VARCHAR2(200) NOT NULL,

"Date" DATE NOT NULL,

Prescription VARCHAR2(200) -- Optional

);

ALTER TABLE MedicalRecords

ADD CONSTRAINT MedicalRecords\_PK PRIMARY KEY (RecordID);

ALTER TABLE MedicalRecords

ADD CONSTRAINT MedicalRecords\_Patients\_FK FOREIGN KEY (PatientID)

REFERENCES Patients (PatientID);

**-- Now, we’ll add 2 more table throughout SQL Developer:**

**-- Insurance**

CREATE TABLE Insurance (

InsuranceID CHAR(6) NOT NULL,

ProviderName VARCHAR2(100) NOT NULL,

PolicyNumber VARCHAR2(50) NOT NULL UNIQUE,

Coverage NUMBER(10,2) CHECK (Coverage >= 0) NOT NULL,

ExpiryDate DATE NOT NULL,

PatientID CHAR(7) NOT NULL,

CONSTRAINT Insurance\_PK PRIMARY KEY (InsuranceID),

CONSTRAINT Insurance\_Patients\_FK FOREIGN KEY (PatientID) REFERENCES Patients (PatientID)

);

CREATE OR REPLACE TRIGGER Check\_ExpiryDate

BEFORE INSERT OR UPDATE ON Insurance

FOR EACH ROW

BEGIN

IF :NEW.ExpiryDate <= SYSDATE THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Expiry date must be in the future.');

END IF;

END;

A screenshot of a computer

Description automatically generated

**-- Prescriptions**

CREATE TABLE Prescriptions (

PrescriptionID CHAR(6) NOT NULL,

RecordID CHAR(6) NOT NULL,

DoctorID CHAR(7) NOT NULL,

PatientID CHAR(7) NOT NULL,

Medication VARCHAR2(200) NOT NULL,

Dosage VARCHAR2(200) NOT NULL,

IssueDate DATE NOT NULL,

CONSTRAINT Prescriptions\_PK PRIMARY KEY (PrescriptionID),

CONSTRAINT Prescriptions\_Records\_FK FOREIGN KEY (RecordID) REFERENCES MedicalRecords (RecordID),

CONSTRAINT Prescriptions\_Doctors\_FK FOREIGN KEY (DoctorID) REFERENCES Doctors (DoctorID),

CONSTRAINT Prescriptions\_Patients\_FK FOREIGN KEY (PatientID) REFERENCES Patients (PatientID)

);

A screenshot of a computer

Description automatically generated

# 5. Data population

**-- Patients (150 records-randomly)**

BEGIN

FOR i IN 1..150 LOOP

INSERT INTO Patients (

PatientID,

PatientName,

BirthDate,

Gender,

Contact

) VALUES (

'P' || LPAD(i, 6, '0'), -- P000001 to P000150

'Patient\_' || i,

TO\_DATE('1930-01-01', 'YYYY-MM-DD') + TRUNC(DBMS\_RANDOM.VALUE(0, 365 \* 95)), -- BirthDate between 1930–2025

CASE

WHEN DBMS\_RANDOM.VALUE < 0.5 THEN 'Male'

ELSE 'Female'

END,

'555-' ||

LPAD(TO\_CHAR(TRUNC(DBMS\_RANDOM.VALUE(100, 999))), 3, '0') || '-' ||

LPAD(TO\_CHAR(TRUNC(DBMS\_RANDOM.VALUE(1000, 9999))), 4, '0') -- Phone format

);

END LOOP;

COMMIT;

END;

/

A screenshot of a computer

Description automatically generated

**-- Doctors:**

**-- You can insert via script like:**

INSERT INTO DOCTORS (DOCTORID, DOCTORNAME, PHONE, WORKINGHOURS)

VALUES ('D000001', 'Dr. Alice Brown', '555-1234', '09:00-17:00');

INSERT INTO DOCTORS (DOCTORID, DOCTORNAME, PHONE, WORKINGHOURS)

VALUES ('D000002', 'Dr. Bob White', '555-5678', '10:00-18:00');

INSERT INTO DOCTORS (DOCTORID, DOCTORNAME, PHONE, WORKINGHOURS)

VALUES ('D000003', 'Dr. Charlie Green', '545-3671', '08:00-14:00');

**-- Or via table (through icon with + )**

A screenshot of a computer

Description automatically generated

**-- MedicalRecords:**

INSERT INTO MEDICALRECORDS (RECORDID, PATIENTID, DIAGNOSIS,"Date", PRESCRIPTION)

VALUES ('MR0001', 'P000001', 'Flu', TO\_DATE('01-JAN-24', 'DD-MON-YY'), 'Paracetamol');

INSERT INTO MEDICALRECORDS (RECORDID, PATIENTID, DIAGNOSIS, "DATE", PRESCRIPTION)

VALUES ('MR0002', 'P000002', 'Cold', TO\_DATE('05-FEB-24', 'DD-MON-YY'), NULL);

INSERT INTO MEDICALRECORDS (RECORDID, PATIENTID, DIAGNOSIS,"Date", PRESCRIPTION)

VALUES ('MR0003', 'P000003', 'Fever', TO\_DATE('10-MAR-24', 'DD-MON-YY'), 'Ibuprofen');

INSERT INTO MEDICALRECORDS (RECORDID, PATIENTID, DIAGNOSIS,"Date", PRESCRIPTION)

VALUES ('MR0004', 'P000004', 'Allergy', TO\_DATE('12-APR-24', 'DD-MON-YY'), 'Antihistamine');

INSERT INTO MEDICALRECORDS (RECORDID, PATIENTID, DIAGNOSIS,"Date", PRESCRIPTION)

VALUES ('MR0005', 'P000005', 'Headache', TO\_DATE('18-MAY-24', 'DD-MON-YY'), 'Aspirin');

INSERT INTO MEDICALRECORDS (RECORDID, PATIENTID, DIAGNOSIS,"Date", PRESCRIPTION)

VALUES ('MR0006', 'P000006', 'Fever', TO\_DATE('12-JUN-24', 'DD-MON-YY'), 'Ibuprofen');

**-- Bills:**

INSERT INTO BILLS (BILLID, AMOUNT\_EURO, PAYMENTSTATUS,"Date", PATIENTID)

VALUES ('B00001', 150.75, 'Unpaid', TO\_DATE('05-JAN-24', 'DD-MON-YY'), 'P000001');

INSERT INTO BILLS (BILLID, AMOUNT\_EURO, PAYMENTSTATUS, "Date", PATIENTID)

VALUES ('B00002', 145.80, 'Pending', TO\_DATE('17-JUN-24', 'DD-MON-YY'), 'P000002');

INSERT INTO BILLS (BILLID, AMOUNT\_EURO, PAYMENTSTATUS,"Date", PATIENTID)

VALUES ('B00003', 75.50, 'Unpaid', TO\_DATE('15-MAR-24', 'DD-MON-YY'), 'P000003');

INSERT INTO BILLS (BILLID, AMOUNT\_EURO, PAYMENTSTATUS,"Date", PATIENTID)

VALUES ('B00004', 300.00, 'Pending', TO\_DATE('20-APR-24', 'DD-MON-YY'), 'P000004');

INSERT INTO BILLS (BILLID, AMOUNT\_EURO, PAYMENTSTATUS,"Date", PATIENTID)

VALUES ('B00005', 125.25, 'Paid', TO\_DATE('25-MAY-24', 'DD-MON-YY'), 'P000005');

INSERT INTO BILLS (BILLID, AMOUNT\_EURO, PAYMENTSTATUS,"Date", PATIENTID)

VALUES ('B00006', 16.00, 'Paid', TO\_DATE('27-FEB-24', 'DD-MON-YY'), 'P000004');

INSERT INTO BILLS (BILLID, AMOUNT\_EURO, PAYMENTSTATUS, "Date", PATIENTID)

VALUES ('B00007', 48.00, 'Pending', TO\_DATE('11-NOV-24', 'DD-MON-YY'), 'P000001');

INSERT INTO BILLS (BILLID, AMOUNT\_EURO, PAYMENTSTATUS,"Date", PATIENTID)

VALUES ('B00008', 89.99, 'Unpaid', TO\_DATE('10-JAN-24', 'DD-MON-YY'), 'P000003');

INSERT INTO BILLS (BILLID, AMOUNT\_EURO, PAYMENTSTATUS, "Date", PATIENTID)

VALUES ('B00009', 190.00, 'Unpaid', TO\_DATE('15-APR-24', 'DD-MON-YY'), 'P000004');

INSERT INTO BILLS (BILLID, AMOUNT\_EURO, PAYMENTSTATUS, "Date", PATIENTID)

VALUES ('B00010', 165.75, 'Paid', TO\_DATE('17-JUN-24', 'DD-MON-YY'), 'P000006');

INSERT INTO BILLS (BILLID, AMOUNT\_EURO, PAYMENTSTATUS,"Date", PATIENTID)

VALUES ('B00011', 167.75, 'Unpaid', TO\_DATE('04-OCT-24', 'DD-MON-YY'), 'P000007');

INSERT INTO BILLS (BILLID, AMOUNT\_EURO, PAYMENTSTATUS,"Date", PATIENTID)

VALUES ('B00012', 25.00, 'Paid', TO\_DATE('04-OCT-24', 'DD-MON-YY'), 'P000006');

INSERT INTO BILLS (BILLID, AMOUNT\_EURO, PAYMENTSTATUS,"Date", PATIENTID)

VALUES ('B00013', 132.00, 'Pending', TO\_DATE('04-NOV-24', 'DD-MON-YY'), 'P000006');

INSERT INTO BILLS (BILLID, AMOUNT\_EURO, PAYMENTSTATUS,"Date", PATIENTID)

VALUES ('B00014', 42.00, 'Unpaid', TO\_DATE('14-NOV-24', 'DD-MON-YY'), 'P000005');

INSERT INTO BILLS (BILLID, AMOUNT\_EURO, PAYMENTSTATUS,"Date", PATIENTID)

VALUES ('B00015', 42.00, 'Unpaid', TO\_DATE('24-DEC-24', 'DD-MON-YY'), 'P000001');

**-- Services:**

INSERT INTO SERVICES (SERVICECODE, SPECIALTY, PRICE\_EURO, DOCTORID)

VALUES ('S00001', 'General Checkup', 50, 'D000001');

INSERT INTO SERVICES (SERVICECODE, SPECIALTY, PRICE\_EURO, DOCTORID)

VALUES ('S00002', 'Cardiology', 200, 'D000002');

INSERT INTO SERVICES (SERVICECODE, SPECIALTY, PRICE\_EURO, DOCTORID)

VALUES ('S00003', 'Dermatology', 75, 'D000003');

**-- Doctor\_Service:**

INSERT INTO DOCTOR\_SERVICE (DOCTORS\_DOCTORID, SERVICES\_SERVICECODE)

VALUES ('D000001', 'S00001');

INSERT INTO DOCTOR\_SERVICE (DOCTORS\_DOCTORID, SERVICES\_SERVICECODE)

VALUES ('D000002', 'S00002');

INSERT INTO DOCTOR\_SERVICE (DOCTORS\_DOCTORID, SERVICES\_SERVICECODE)

VALUES ('D000003', 'S00003');

**-- Appointments:**

INSERT INTO APPOINTMENTS (APPOINTMENTID, DATETIME, VISITREASON, PATIENTID, DOCTORID)

VALUES ('A000001', TO\_DATE('05-JAN-24', 'DD-MON-YY'), 'Routine Checkup', 'P000001', 'D000001');

INSERT INTO APPOINTMENTS (APPOINTMENTID, DATETIME, VISITREASON, PATIENTID, DOCTORID)

VALUES ('A000002', TO\_DATE('15-FEB-24', 'DD-MON-YY'), 'Heart Checkup', 'P000002', 'D000002');

INSERT INTO APPOINTMENTS (APPOINTMENTID, DATETIME, VISITREASON, PATIENTID, DOCTORID)

VALUES ('A000003', TO\_DATE('20-MAR-24', 'DD-MON-YY'), 'Skin Allergy', 'P000003', 'D000003');

INSERT INTO APPOINTMENTS

VALUES ('A000004', TO\_DATE('18-JUN-24', 'DD-MON-YY'), 'Heart Surgery', 'P000004', 'D000002');

**-- Insurance:**

INSERT INTO INSURANCE (INSURANCEID, PROVIDERNAME, POLICYNUMBER, COVERAGE, EXPIRYDATE, PATIENTID)

VALUES ('INS001', 'Blue Cross Blue Shield', 'BCBS123456', 5000, TO\_DATE('31-DEC-25', 'DD-MON-YY'), 'P000001');

INSERT INTO INSURANCE (INSURANCEID, PROVIDERNAME, POLICYNUMBER, COVERAGE, EXPIRYDATE, PATIENTID)

VALUES ('INS002', 'UnitedHealthcare', 'UHC789012', 7500, TO\_DATE('15-NOV-27', 'DD-MON-YY'), 'P000002');

INSERT INTO INSURANCE (INSURANCEID, PROVIDERNAME, POLICYNUMBER, COVERAGE, EXPIRYDATE, PATIENTID)

VALUES ('INS003', 'Aetna', 'AET345678', 10000, TO\_DATE('20-OCT-26', 'DD-MON-YY'), 'P000003');

INSERT INTO INSURANCE (INSURANCEID, PROVIDERNAME, POLICYNUMBER, COVERAGE, EXPIRYDATE, PATIENTID)

VALUES ('INS004', 'Cigna', 'CIG901234', 3000, TO\_DATE('30-SEP-27', 'DD-MON-YY'), 'P000004');

**-- Prescriptions:**

INSERT INTO PRESCRIPTIONS (PRESCRIPTIONID, RECORDID, DOCTORID, PATIENTID, MEDICATION, DOSAGE, ISSUEDATE)

VALUES ('RX0006', 'MR0005', 'D000001', 'P000005', 'Ibuprofen', '200mg', TO\_DATE('05-JUN-24', 'DD-MON-YY'));

INSERT INTO PRESCRIPTIONS (PRESCRIPTIONID, RECORDID, DOCTORID, PATIENTID, MEDICATION, DOSAGE, ISSUEDATE)

VALUES ('RX0001', 'MR0001', 'D000001', 'P000001', 'Paracetamol', '500mg', TO\_DATE('01-JAN-24', 'DD-MON-YY'));

INSERT INTO PRESCRIPTIONS (PRESCRIPTIONID, RECORDID, DOCTORID, PATIENTID, MEDICATION, DOSAGE, ISSUEDATE)

VALUES ('RX0003', 'MR0003', 'D000003', 'P000003', 'Ibuprofen', '400mg', TO\_DATE('10-MAR-24', 'DD-MON-YY'));

INSERT INTO PRESCRIPTIONS (PRESCRIPTIONID, RECORDID, DOCTORID, PATIENTID, MEDICATION, DOSAGE, ISSUEDATE)

VALUES ('RX0004', 'MR0004', 'D000003', 'P000004', 'Antihistamine', '20mg', TO\_DATE('12-APR-24', 'DD-MON-YY'));

INSERT INTO PRESCRIPTIONS (PRESCRIPTIONID, RECORDID, DOCTORID, PATIENTID, MEDICATION, DOSAGE, ISSUEDATE)

VALUES ('RX0005', 'MR0005', 'D000001', 'P000005', 'Aspirin', '325mg', TO\_DATE('18-MAY-24', 'DD-MON-YY'));

**-- Index creation**

/\*Different columns can be chosen for indexing, but because we have 150 patients- each may have different prescriptions which make searching very slow. I indexed patient id on prescription table, because usually hospitals&pharmacies use patient id (because it is unique, easier than name) for searching for prescriptions./\*

Create Index idx\_patientid ON Prescriptions(PatientID);

# 6. PL/SQL (beginner to advanced)

**--1. Identifies all medications (without repetition) used to treat fever**

SELECT DISTINCT PRESCRIPTION

FROM MEDICALRECORDS

WHERE DIAGNOSIS = 'Fever';

A screenshot of a computer

Description automatically generated

**-- 2. Find doctors whose phone numbers end with '71'**

SELECT DoctorID, DoctorName, Phone

FROM Doctors

WHERE Phone LIKE '%71';

A screenshot of a computer

Description automatically generated

**-- 3. Search elderly male patients born before 1965 and phone number with ‘555-6’:**

SELECT \*

FROM PATIENTS

WHERE GENDER = 'Male' AND TO\_DATE(BIRTHDATE, 'DD-MON-YY') < TO\_DATE('01-JAN-65', 'DD-MON-YY') AND CONTACT LIKE '555-6%';

A screenshot of a computer

Description automatically generated

**-- 4. Find prescriptions in which the prescription date is earlier than 15 April 2024 AND medication dosage is higher than 300 mg or Antihistamine medication**

SELECT PRESCRIPTIONID, MEDICATION, DOSAGE, ISSUEDATE

FROM PRESCRIPTIONS

WHERE TO\_DATE(ISSUEDATE, 'DD-MON-YY') < TO\_DATE('15-APR-24', 'DD-MON-YY')

AND (TO\_NUMBER(SUBSTR(DOSAGE, 1, INSTR(DOSAGE, 'mg')-1))) > 300

OR MEDICATION LIKE '%Antihistamine%';

A screenshot of a computer

Description automatically generated

**-- 5. Find how many bills you have in Paid and Pending categories and total amount of payment –income by these categories (since pending means partially paid)**

SELECT

PAYMENTSTATUS,

SUM(AMOUNT\_EURO) AS "Total Amount",

COUNT(BILLID) AS "Number of Bills"

FROM BILLS

WHERE PAYMENTSTATUS IN ('Paid', 'Pending')

GROUP BY PAYMENTSTATUS

ORDER BY "Total Amount" DESC;

A white screen with black text

Description automatically generated

**-- 6. Which doctors had more than 1 patient visit:**

SELECT

DOCTORID,

COUNT(PATIENTID) AS "Patient Visits"

FROM PRESCRIPTIONS

GROUP BY DOCTORID

HAVING COUNT(PATIENTID) > 1;

A screenshot of a computer

Description automatically generated

**-- 7. Displays PAYMENTSTATUS (Paid, Unpaid, Pending) having the sum of all bills for that status is between 250 and 500 euro:**

SELECT PAYMENTSTATUS, SUM(AMOUNT\_EURO) AS "Total Amount"

FROM BILLS

GROUP BY PAYMENTSTATUS

HAVING SUM(AMOUNT\_EURO) BETWEEN 250 AND 500;

A screenshot of a computer

Description automatically generated

**-- 8. This query identifies diagnoses that were treated with medications in 2 or more cases:**

SELECT

DIAGNOSIS,

COUNT(RECORDID) AS "Case\_Count"

FROM MEDICALRECORDS

WHERE PRESCRIPTION IS NOT NULL

GROUP BY DIAGNOSIS

HAVING COUNT(RECORDID) >= 2;

A screenshot of a computer

Description automatically generated

**--9. Shows only patients that have matching bill records**

SELECT P.PATIENTID, P.PATIENTNAME, b.BILLID, b.AMOUNT\_EURO, b.PAYMENTSTATUS, b."Date" AS BILL\_DATE

FROM PATIENTS p

JOIN BILLS b ON p.PATIENTID = b.PATIENTID

ORDER BY b."Date" DESC;

A screenshot of a computer

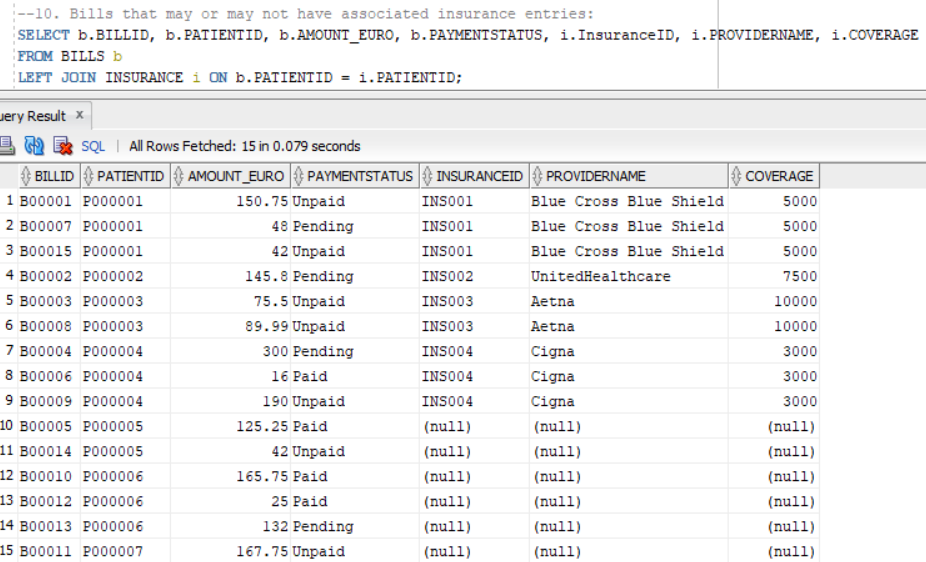
Description automatically generated

**--10. Bills that may or may not have associated insurance entries:**

SELECT b.BILLID, b.PATIENTID, b.AMOUNT\_EURO, b.PAYMENTSTATUS, i.InsuranceID, i.PROVIDERNAME, i.COVERAGE

FROM BILLS b

LEFT JOIN INSURANCE i ON b.PATIENTID = i.PATIENTID;



**--11. Creates a human-readable schedule by mentioning patient and doctor names clearly on appointments**

SELECT a.APPOINTMENTID, P.PATIENTNAME, d.DOCTORNAME, a.VISITREASON, a.DATETIME

FROM APPOINTMENTS a

JOIN PATIENTS P ON a.PATIENTID = p.PATIENTID

JOIN DOCTORS d ON a.DOCTORID = d.DOCTORID;

A screenshot of a computer

Description automatically generated

**--12. How many prescriptions each doctor issued to each patient**

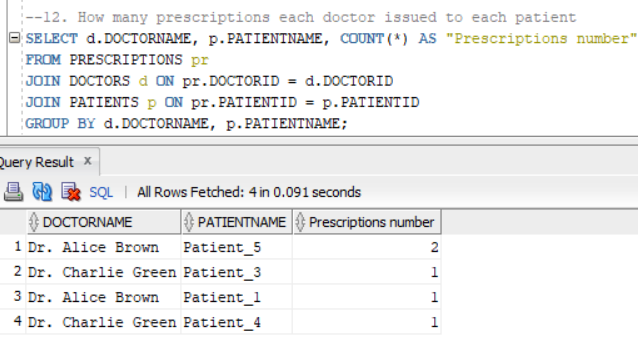
SELECT d.DOCTORNAME, p.PATIENTNAME, COUNT(\*) AS "Prescriptions number"

FROM PRESCRIPTIONS pr

JOIN DOCTORS d ON pr.DOCTORID = d.DOCTORID

JOIN PATIENTS p ON pr.PATIENTID = p.PATIENTID

GROUP BY d.DOCTORNAME, p.PATIENTNAME;



**--13. Retrieve all patients who have unpaid bills.**

SELECT PATIENTID, PATIENTNAME, CONTACT

FROM PATIENTS

WHERE PATIENTID IN (

SELECT PATIENTID

FROM BILLS

WHERE PAYMENTSTATUS = 'Unpaid'

);

A screenshot of a computer

Description automatically generated

**-- 14. Doctors who have more appointments than the average number of appointments per doctor:**

SELECT d.DOCTORNAME, COUNT(a.APPOINTMENTID) AS AppointmentCount

FROM DOCTORS d

JOIN APPOINTMENTS a ON d.DOCTORID = a.DOCTORID

GROUP BY d.DOCTORNAME

HAVING COUNT(a.APPOINTMENTID) > (

SELECT AVG(AppointmentCount)

FROM (

SELECT COUNT(APPOINTMENTID) AS AppointmentCount

FROM APPOINTMENTS

GROUP BY DOCTORID

)

);

A screenshot of a computer program

Description automatically generated

**-- 15. What percentage each insurance provider's coverage represents out of the toal covarage across all providers to compare market share:**

SELECT

PROVIDERNAME,

COVERAGE,

ROUND(COVERAGE \* 100 / total\_coverage.total, 2) as percentage

FROM INSURANCE,

(SELECT SUM(COVERAGE) as total FROM INSURANCE) total\_coverage

ORDER BY percentage DESC;

A screenshot of a computer

Description automatically generated

**-- 16. Prescription patterns using ROLLUP to show:**

**-- Counts for each specific medication+dosage combination**

**-- Subtotals for each medication (all dosages combined)**

**-- Grand total of all prescriptions**

SELECT

MEDICATION,

DOSAGE,

COUNT(\*) as prescription\_count

FROM PRESCRIPTIONS

GROUP BY ROLLUP(MEDICATION, DOSAGE)

ORDER BY MEDICATION, DOSAGE;

A screenshot of a computer

Description automatically generated

**-- 17. Appointment counts by doctor and distributed across months**

SELECT \* FROM (

SELECT

DOCTORID,

PATIENTID,

TO\_CHAR(DATETIME, 'YYYY-MM') as month

FROM APPOINTMENTS

)

PIVOT (

COUNT(PATIENTID)

FOR month IN (

'2024-01' as Jan,

'2024-02' as Feb,

'2024-03' as Mar,

'2024-04' as Apr,

'2024-05' as May

)

)

ORDER BY DOCTORID;

A screenshot of a calendar

Description automatically generated

**-- 18. Sets insurance coverage amounts for the next two years assuming a 10% annual increase**

**-- Ex: If Blue Cross has €5000 coverage in 2024, what will it be in 2026 with 10% yearly growth?**

SELECT PROVIDERNAME, YEAR, COVERAGE

FROM (

SELECT PROVIDERNAME, 2024 as YEAR, COVERAGE

FROM INSURANCE

)

MODEL

PARTITION BY (PROVIDERNAME)

DIMENSION BY (YEAR)

MEASURES (COVERAGE)

RULES (

COVERAGE[2025] = COVERAGE[2024] \* 1.1,

COVERAGE[2026] = COVERAGE[2025] \* 1.1

)

ORDER BY PROVIDERNAME, YEAR;

A screenshot of a table

Description automatically generated

**--19. PL/SQL Function: The total amount of unpaid bills for a specific patient**

CREATE OR REPLACE FUNCTION get\_unpaid\_bills\_total(

p\_patient\_id IN CHAR

) RETURN NUMBER IS

v\_total\_amount NUMBER(10,2);

BEGIN

SELECT SUM(Amount\_Euro)

INTO v\_total\_amount

FROM Bills

WHERE PatientID = p\_patient\_id

AND PaymentStatus = 'Unpaid';

RETURN NVL(v\_total\_amount, 0);

EXCEPTION

WHEN OTHERS THEN

RETURN -1;

END get\_unpaid\_bills\_total;

/

**-- Test the function**

DECLARE

v\_result NUMBER;

BEGIN

v\_result := get\_unpaid\_bills\_total('P000003');

DBMS\_OUTPUT.PUT\_LINE('Total unpaid bills for P000003: ' || v\_result || ' Euro');

v\_result := get\_unpaid\_bills\_total('P000001');

DBMS\_OUTPUT.PUT\_LINE('Total unpaid bills for P000001: ' || v\_result || ' Euro');

v\_result := get\_unpaid\_bills\_total('P000004');

DBMS\_OUTPUT.PUT\_LINE('Total unpaid bills for P000004: ' || v\_result || ' Euro');

END;

/

A screenshot of a computer

Description automatically generated

**-- 20. Procedure for patients with unpaid bills**

CREATE OR REPLACE PROCEDURE list\_patients\_with\_unpaid\_bills IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Patients with unpaid bills:');

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

FOR patient\_rec IN (

SELECT p.PatientID, p.PatientName,

get\_unpaid\_bills\_total(p.PatientID) AS UnpaidTotal -- **(a) Uses the function from Task 19**

FROM Patients p

WHERE get\_unpaid\_bills\_total(p.PatientID) > 0

)

LOOP

**-- (b) Query retrieves multiple rows; results are printed**

DBMS\_OUTPUT.PUT\_LINE(

'Patient ID: ' || patient\_rec.PatientID ||

', Name: ' || patient\_rec.PatientName ||

', Unpaid Total: ' || patient\_rec.UnpaidTotal || ' Euro'

);

END LOOP;

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

DBMS\_OUTPUT.PUT\_LINE('End of report');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error in list\_patients\_with\_unpaid\_bills: ' || SQLERRM);

END list\_patients\_with\_unpaid\_bills;

/

BEGIN

list\_patients\_with\_unpaid\_bills();

END;

/

A screenshot of a computer

Description automatically generated

**-- 21. Trigger for bill status before and after an update**

CREATE OR REPLACE TRIGGER update\_payment\_status

BEFORE UPDATE OF AMOUNT\_EURO ON BILLS

FOR EACH ROW

BEGIN

**-- If the amount is being set to 0, automatically mark the bill as paid**

IF :NEW.AMOUNT\_EURO = 0 AND :OLD.AMOUNT\_EURO != 0 THEN

:NEW.PAYMENTSTATUS := 'Paid';

END IF;

END;

/

BEGIN

**-- Display status before update**

DBMS\_OUTPUT.PUT\_LINE('Before update:');

FOR bill\_rec IN (

SELECT BILLID, AMOUNT\_EURO, PAYMENTSTATUS

FROM BILLS

WHERE BILLID = 'B00003'

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Bill ' || bill\_rec.BILLID || ': ' || bill\_rec.AMOUNT\_EURO || ' Euro, Status: ' || bill\_rec.PAYMENTSTATUS);

END LOOP;

**-- Update the amount to 0 — this should activate the trigger**

UPDATE BILLS SET AMOUNT\_EURO = 0 WHERE BILLID = 'B00003';

**-- Display status after update**

DBMS\_OUTPUT.PUT\_LINE('After update:');

FOR bill\_rec IN (

SELECT BILLID, AMOUNT\_EURO, PAYMENTSTATUS

FROM BILLS

WHERE BILLID = 'B00003'

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Bill ' || bill\_rec.BILLID || ': ' || bill\_rec.AMOUNT\_EURO || ' Euro, Status: ' || bill\_rec.PAYMENTSTATUS);

END LOOP;

ROLLBACK;

END;

/

A screenshot of a computer

Description automatically generated