HEALTH CARE MANAGEMENT SYSTEM

BUAN 6320 – Database Foundations for Business Analytics

PROJECT – TECHNICAL REPORT

Group- 6

Don Thoppil Jain

Sushma Bukkapuram

Tara Canugovi

Mounika Kolle

Venkata Lakshmi Ishwarya Tatavarthi

Table of Contents:

Introduction:

This database design document describes the requirements and defines the Entity Relationship Diagram (ERD) for a Health Care Management System database. The system enables patients to book an appointment with the doctor and have a track of lab test and medical records.

Requirement Definition Details:

Entity and Attribute description

Here's a description of each entity and its attributes from the healthcare management system ER diagram:

1-Patients:

Description: This entity represents the individuals receiving medical care within the healthcare system.

Attributes:

- PatientID: Unique identifier for each patient.
- Name: Full name of the patient.
- DateOfBirth: Birthdate of the patient.
- Gender: Gender of the patient.
- ContactInfo: Contact details, such as phone number or email.
- Address: Home address of the patient.
- MedicalHistory: Summary of the patient's past medical history.

2-Doctor:

Description: Represents the healthcare providers who diagnose and treat patients.

Attributes:

- DoctorID: Unique identifier for each doctor.
- Name: Full name of the doctor.
- Department: Department in which the doctor works.
- Specialty: Medical specialty of the doctor.
- Experience: Years of experience the doctor has.
- ContactInfo: Contact information, such as phone number or email.

3. Appointment:

Description: Tracks the scheduled meetings between patients and doctors.

Attributes:

- AppointmentID: Unique identifier for each appointment.
- PatientID (FK): Foreign key linking to the patient attending the appointment.
- DoctorID (FK): Foreign key linking to the doctor assigned to the appointment.
- Date: Date and time of the appointment.
- AppointmentReason: Reason for the appointment.
- Status: Status of the appointment (e.g., scheduled, completed, canceled).

4. Billing:

Description: Records of payment information related to patient appointments.

Attributes:

- BillID: Unique identifier for each billing record.
- PatientID (FK): Foreign key linking to the patient associated with the bill.
- AppointmentID (FK): Foreign key linking to the appointment that generated the bill.
- TotalAmount: Total cost for the appointment.
- Date: Date the bill was generated.
- PaymentStatus: Status of the payment (e.g., paid, pending).
- InsuranceDetails: Information on insurance coverage, if applicable.

5. MedicalRecord:

Description: Documents of the medical details and treatment history of patients.

Attributes:

- RecordID: Unique identifier for each medical record.
- PatientID (FK): Foreign key linking to the patient to whom the record belongs.
- AppointmentID (FK): Foreign key linking to the appointment associated with the record.
- DoctorID (FK): Foreign key linking to the doctor responsible for the record.
- Notes: Doctor's notes regarding the patient's condition.
- Treatment: Description of treatments provided.

6. Lab_Tests:

Description: Represents various lab tests conducted as part of a patient's medical care.

Attributes:

- TestID: Unique identifier for each lab test.
- RecordID (FK): Foreign key linking to the medical record associated with the test.
- TestName: Name of the lab test (e.g., blood test, MRI).
- Attribute4: Additional attribute, potentially used for specific test characteristics.
- TestDate: Date the test was performed.
- Result: Result or outcome of the lab test.
- DoctorNotes: Notes from the doctor regarding the test results.
- Each entity in this system has attributes to identify the key information required for managing patient care, medical records, and billing in a healthcare setting.

Relationship and Cardinality description:

Patients - Appointments

- Relationship: A patient can schedule one or more appointments.
- Cardinality: 1:M from Patients to Appointments.
- Description: Each patient can have multiple appointments, but each appointment is linked to one specific patient.

Doctors - Appointments

- Relationship: A doctor can have one or more appointments with different patients.
- Cardinality: 1:M from Doctors to Appointments.
- Description: Each doctor can be assigned multiple appointments, but each appointment is assigned to only one doctor.

Patients – MedicalRecords

- Relationship: Each patient can have multiple medical records.
- Cardinality: 1:M from Patients to MedicalRecords.
- Description: A patient may have multiple medical records over time, each corresponding to an appointment or treatment.

Appointments – MedicalRecords

- Relationship: Each appointment generates a medical record.
- Cardinality: 1:1 from Appointments to MedicalRecords.
- Description: Each appointment corresponds to one unique medical record, capturing the treatment and diagnosis.

Doctors - MedicalRecords

- Relationship: A doctor contributes to multiple medical records for different patients.
- Cardinality: 1:M from Doctors to MedicalRecords.
- Description: Each doctor may be associated with multiple medical records if they treated multiple patients, but each medical record is linked to one specific doctor.

MedicalRecords - LabTests

- Relationship: Each medical record can have one or more lab tests associated with it.
- Cardinality: 1:M from MedicalRecords to LabTests.
- Description: Each medical record may involve multiple lab tests conducted as part of the patient's treatment or diagnosis.

Patients - Billing

- Relationship: Each patient can have multiple bills.
- Cardinality: 1:M from Patients to Billing.
- Description: A patient can have multiple bills generated for different appointments or treatments, but each bill corresponds to one specific patient.

Appointments – Billing

- Relationship: Each appointment generates a bill.
- Cardinality: 1:1 from Appointments to Billing.
- Description: Each appointment has a single bill associated with it, capturing the cost of services provided in that session.

Assumptions and Special Considerations:

1. Patient Registration: It is assumed that each patient must be registered in the system before scheduling appointments.
2. Unique Identifiers: Each patient, doctor, appointment, and medical record has a unique identifier to ensure accurate data tracking.
3. Standardized Contact Information: Patient and doctor contact information (e.g., phone numbers) is stored in a standardized format.
4. Insurance Information: It is assumed that billing records may include insurance details, with payment status updated as claims are processed.
5. Role-Based Access Control: Access to sensitive patient data is limited based on user roles (e.g., doctors, billing staff) to ensure data privacy and compliance.

These assumptions and considerations ensure data accuracy, privacy, and compliance in the healthcare management system.

Normalization:

To demonstrate the normalization process of the hospital database schema up to the Third Normal Form (3NF), let's go through each step, identifying and resolving dependencies and documenting how we address any transitive dependencies.

Step 1: First Normal Form (1NF)

Definition: In 1NF, each table should contain only atomic values (no repeating groups or arrays).

Current Schema:

- Patients: PatientID, Name, DateOfBirth, Gender, ContactInfo, Address, MedicalHistory
- Doctors: DoctorID, Name, Specialty, ContactInfo, Department, Experience
- Appointments: AppointmentID, PatientID (FK), DoctorID (FK), Date, Time, AppointmentReason, Status
- MedicalRecords: RecordID, PatientID (FK), AppointmentID (FK), DoctorID (FK), Diagnosis, Treatment, Notes
- LabTests: TestID, RecordID (FK), TestName, TestDate, Result, DoctorNotes
- Billing: BillID, PatientID (FK), AppointmentID (FK), TotalAmount, PaymentStatus, Date, InsuranceDetails

Analysis: Each attribute in these tables is already atomic (i.e., single-valued), so the schema satisfies 1NF.

Step 2: Second Normal Form (2NF)

Definition: In 2NF, the schema must be in 1NF, and all non-key attributes must be fully dependent on the primary key (no partial dependencies).

Current Schema Analysis:

- Each table has a single primary key: PatientID for Patients, DoctorID for Doctors, AppointmentID for Appointments, RecordID for MedicalRecords, TestID for LabTests, and BillID for Billing.
- In each table, all non-key attributes depend fully on their respective primary keys, so there are no partial dependencies.

Conclusion: Each table satisfies 2NF.

Step 3: Third Normal Form (3NF)

Definition: In 3NF, the schema must be in 2NF, and there should be no transitive dependencies (i.e., non-key attributes should not depend on other non-key attributes).

Identifying and Resolving Transitive Dependencies: To demonstrate resolving a transitive dependency, let's focus on an example within the **Billing** table.

Billing Table Example

Attributes: BillID, PatientID (FK), AppointmentID (FK), TotalAmount, PaymentStatus, Date, InsuranceDetails

Identified Transitive Dependency: Suppose that InsuranceDetails depends on PatientID rather than BillID, indicating that InsuranceDetails might be better associated with the patient rather than with each bill.

Solution:

1. Remove InsuranceDetails from the Billing table.
2. Create a new table, **PatientInsurance**, which associates insurance details directly with each patient.

Revised Billing and New PatientInsurance Tables:

- **Billing**: BillID, PatientID (FK), AppointmentID (FK), TotalAmount, PaymentStatus, Date
- **PatientInsurance**: PatientID, InsuranceDetails

Updated Schema Summary in 3NF:

- Patients: PatientID, Name, DateOfBirth, Gender, ContactInfo, Address, MedicalHistory
- Doctors: DoctorID, Name, Specialty, ContactInfo, Department, Experience
- Appointments: AppointmentID, PatientID (FK), DoctorID (FK), Date, Time, AppointmentReason, Status
- MedicalRecords: RecordID, PatientID (FK), AppointmentID (FK), DoctorID (FK), Diagnosis, Treatment, Notes
- LabTests: TestID, RecordID (FK), TestName, TestDate, Result, DoctorNotes
- Billing: BillID, PatientID (FK), AppointmentID (FK), TotalAmount, PaymentStatus, Date
- PatientInsurance: PatientID, InsuranceDetails

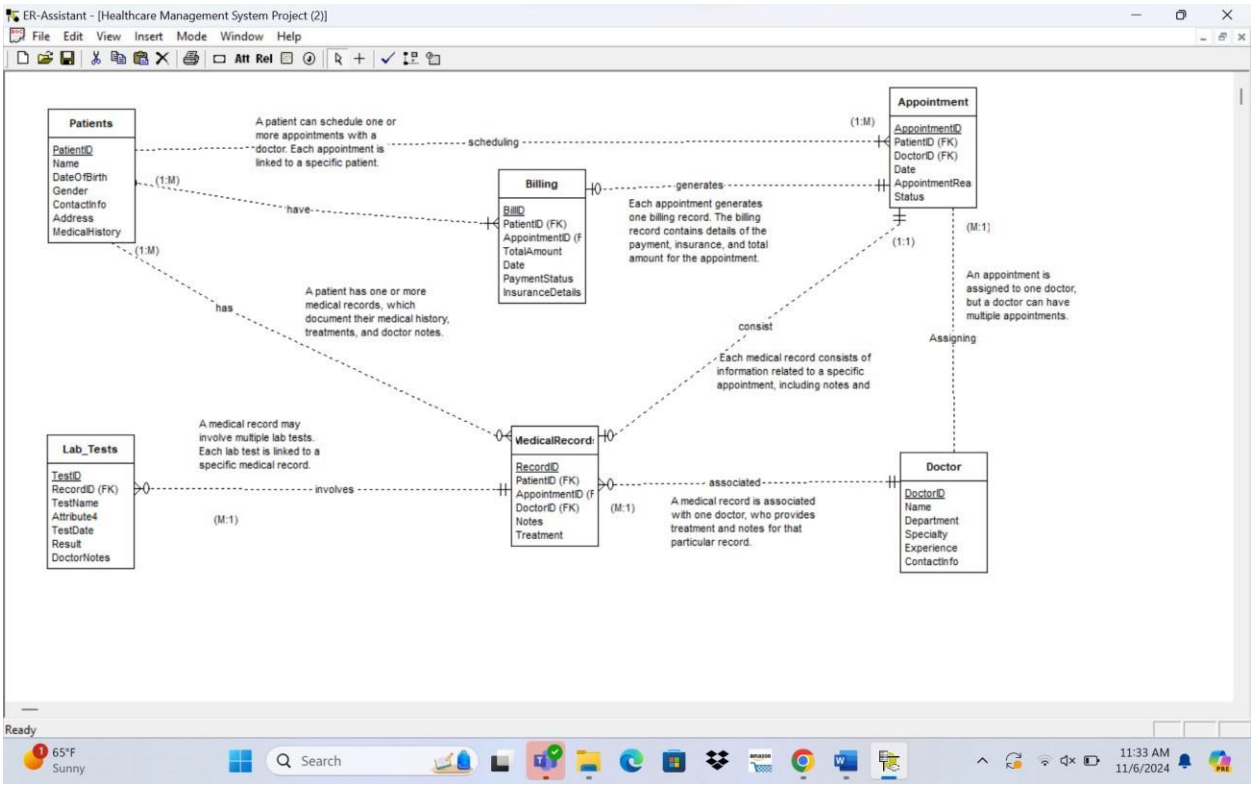
Summary of Normalization Process

1. **1NF**: Ensured all attributes are atomic in each table.
2. **2NF**: Verified that all non-key attributes are fully dependent on the primary key in each table.
3. **3NF**: Identified a transitive dependency in the **Billing** table (InsuranceDetails depending on PatientID rather than BillID). Resolved this by moving InsuranceDetails to a new **PatientInsurance** table, making the schema fully normalized in 3NF.

This completes the normalization process, reducing redundancy and ensuring data integrity by organizing attributes around primary keys and eliminating dependencies on non-key attributes.

Entity Relationship Diagram (ERD):

The following is the Food Delivery System database based on the requirement definition details highlighting the entity, attributes, relationships and cardinality details:



Technical Report:

I. Data Definition Language (DDL)

```
set search_path to healthcare_system;


-- Drop Triggers
DROP TRIGGER IF EXISTS assign_bill_id_trigger ON Billing;
DROP TRIGGER IF EXISTS assign_test_id_trigger ON Lab_Tests;
DROP TRIGGER IF EXISTS assign_record_id_trigger ON MedicalRecord;
DROP TRIGGER IF EXISTS assign_appointment_id_trigger ON Appointment;
DROP TRIGGER IF EXISTS assign_doctor_id ON Doctor;
DROP TRIGGER IF EXISTS assign_patient_id_trigger ON Patients;

-- Drop Functions
DROP FUNCTION IF EXISTS assign_patient_id();
DROP FUNCTION IF EXISTS assign_doctor_id();
DROP FUNCTION IF EXISTS assign_appointment_id();
DROP FUNCTION IF EXISTS assign_record_id();
DROP FUNCTION IF EXISTS assign_test_id();
DROP FUNCTION IF EXISTS assign_bill_id();

-- Drop Sequences
DROP SEQUENCE IF EXISTS patient_id_seq;
DROP SEQUENCE IF EXISTS doctor_id_seq;
DROP SEQUENCE IF EXISTS appointment_id_seq;
DROP SEQUENCE IF EXISTS record_id_seq;
DROP SEQUENCE IF EXISTS test_id_seq;
DROP SEQUENCE IF EXISTS bill_id_sequen;

----Views-------------
DROP VIEW IF EXISTS PatientAppointmentsView;
DROP VIEW IF EXISTS DoctorAppointmentsView;
DROP VIEW IF EXISTS BillingSummaryView;
DROP VIEW IF EXISTS LabTestDetailsView;

-- Drop indices for Patients
DROP INDEX IF EXISTS idx_patients_name;
DROP INDEX IF EXISTS idx_patients_name_dob;

-- Drop indices for Doctor
DROP INDEX IF EXISTS idx_doctor_specialty;
DROP INDEX IF EXISTS idx_doctor_department_specialty;

-- Drop indices for Appointment
DROP INDEX IF EXISTS idx_appointment_patient_id;
DROP INDEX IF EXISTS idx_appointment_doctor_id;
```

```
DROP INDEX IF EXISTS idx_appointment_date;



-- Drop indices for Billing
DROP INDEX IF EXISTS idx_billing_patient_id;
DROP INDEX IF EXISTS idx_billing_appointment_id;
DROP INDEX IF EXISTS idx_billing_payment_status;

-- Drop indices for MedicalRecord
DROP INDEX IF EXISTS idx_medicalrecord_patient_id;
DROP INDEX IF EXISTS idx_medicalrecord_appointment_id;
DROP INDEX IF EXISTS idx_medicalrecord_doctor_id;

-- Drop indices for Lab_Tests
DROP INDEX IF EXISTS idx_labtests_record_id;
DROP INDEX IF EXISTS idx_labtests_test_date;

-- Drop Lab_Tests table
DROP TABLE IF EXISTS Lab_Tests;

-- Drop MedicalRecord table
DROP TABLE IF EXISTS MedicalRecord;



-- Drop Billing table
DROP TABLE IF EXISTS Billing;

-- Drop Appointment table
DROP TABLE IF EXISTS Appointment;

-- Drop Doctor table
DROP TABLE IF EXISTS Doctor;

-- Drop Patients table
DROP TABLE IF EXISTS Patients;

-----Table creations
CREATE TABLE Patients (
    PatientID SERIAL PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    DateOfBirth DATE NOT NULL
);

CREATE TABLE Doctor (
    DoctorID SERIAL PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Department VARCHAR(100) NOT NULL
);
```

```
CREATE TABLE Appointment (
   AppointmentID INT PRIMARY KEY,
   PatientID INT REFERENCES Patients(PatientID),
   DoctorID INT REFERENCES Doctor(DoctorID)
);
CREATE TABLE MedicalRecord (
   RecordID INT PRIMARY KEY,
   PatientID INT REFERENCES Patients(PatientID),
   AppointmentID INT REFERENCES Appointment(AppointmentID)
);

CREATE TABLE Lab_Tests (
   TestID INT PRIMARY KEY,
   RecordID INT REFERENCES MedicalRecord(RecordID),
   TestName VARCHAR(100)
);


CREATE TABLE Billing (
   BillID INT PRIMARY KEY,
   PatientID INT REFERENCES Patients(PatientID),
   AppointmentID INT REFERENCES Appointment(AppointmentID)
);


ALTER TABLE Billing
ADD TotalAmount NUMERIC(10, 2),
ADD Date DATE,
ADD PaymentStatus VARCHAR(20),
ADD InsuranceDetails TEXT;

ALTER TABLE Lab_Tests
ADD TestDate DATE NOT NULL,
ADD Result TEXT NOT NULL,
ADD DoctorNotes TEXT;

ALTER TABLE MedicalRecord
ADD DoctorID INT REFERENCES Doctor(DoctorID),
ADD Notes TEXT,
ADD Treatment TEXT NOT NULL;

ALTER TABLE Appointment
ADD Date DATE NOT NULL,
ADD AppointmentReason TEXT,
ADD Status VARCHAR(20) NOT NULL;

ALTER TABLE Doctor
ADD Specialty VARCHAR(100),
ADD Experience INT NOT NULL,
```

```
ADD ContactInfo VARCHAR(255);

ALTER TABLE Patients
ADD Gender CHAR(1) CHECK (Gender IN ('M', 'F', 'O')),
ADD ContactInfo VARCHAR(255),
ADD Address VARCHAR(255),
ADD MedicalHistory TEXT;
-- Indices for Patients
CREATE INDEX idx_patients_name ON Patients (Name);
CREATE INDEX idx_patients_name_dob ON Patients (Name, DateOfBirth);

-- Indices for Doctor
CREATE INDEX idx_doctor_specialty ON Doctor (Specialty);
CREATE INDEX idx_doctor_department_specialty ON Doctor (Department, Specialty);

-- Indices for Appointment
CREATE INDEX idx_appointment_patient_id ON Appointment (PatientID);
CREATE INDEX idx_appointment_doctor_id ON Appointment (DoctorID);
CREATE INDEX idx_appointment_date ON Appointment (Date);

-- Indices for Billing
CREATE INDEX idx_billing_patient_id ON Billing (PatientID);
CREATE INDEX idx_billing_appointment_id ON Billing (AppointmentID);
CREATE INDEX idx_billing_payment_status ON Billing (PaymentStatus);

-- Indices for MedicalRecord
CREATE INDEX idx_medicalrecord_patient_id ON MedicalRecord (PatientID);
CREATE INDEX idx_medicalrecord_appointment_id ON MedicalRecord (AppointmentID); CREATE
INDEX idx_medicalrecord_doctor_id ON MedicalRecord (DoctorID);

-- Indices for Lab_Tests
CREATE INDEX idx_labtests_record_id ON Lab_Tests (RecordID);
CREATE INDEX idx_labtests_test_date ON Lab_Tests (TestDate);



CREATE OR REPLACE VIEW PatientAppointmentsView AS
SELECT
   p.PatientID,
   p.Name AS PatientName,
   p.DateOfBirth,
   a.AppointmentID,
   a.Date AS AppointmentDate,
   a.AppointmentReason,
   a.Status
FROM
   Patients p
JOIN
   Appointment a ON p.PatientID = a.PatientID
```

```
CREATE OR REPLACE VIEW DoctorAppointmentsView AS
SELECT
   d.DoctorID,
   d.Name AS DoctorName,
   d.Specialty,
   d.Experience,
   a.AppointmentID,
   a.Date AS AppointmentDate,
   a.AppointmentReason,
   a.Status
FROM
   Doctor d
JOIN
   Appointment a ON d.DoctorID = a.DoctorID;


CREATE OR REPLACE VIEW BillingSummaryView AS
SELECT
   p.PatientID,
   p.Name AS PatientName,
   a.AppointmentID,
   b.BillID,
   b.TotalAmount,
   b.Date AS BillingDate,
   b.PaymentStatus,
   b.InsuranceDetails
FROM
   Patients p
JOIN
   Appointment a ON p.PatientID = a.PatientID
JOIN
   Billing b ON a.AppointmentID = b.AppointmentID;

CREATE OR REPLACE VIEW LabTestDetailsView AS
SELECT
   m.RecordID,
   p.PatientID,
   p.Name AS PatientName,
   l.TestID,
   l.TestName,
   l.TestDate,
   l.Result,
   l.DoctorNotes
FROM
   MedicalRecord m
JOIN
   Patients p ON m.PatientID = p.PatientID
JOIN
   Lab_Tests l ON m.RecordID = l.RecordID;
```

```
CREATE SEQUENCE patient_id_seq
   START WITH 1
   INCREMENT BY 1
   NO MINVALUE
   NO MAXVALUE
   CACHE 1;

CREATE SEQUENCE doctor_id_seq
          START WITH 100
          INCREMENT BY 1
          NO MINVALUE
          NO MAXVALUE
          CACHE 1;

CREATE SEQUENCE appointment_id_seq
          START WITH 1000
          INCREMENT BY 1;

CREATE SEQUENCE record_id_seq
   START WITH 3000
   INCREMENT BY 1
   NO MINVALUE
   NO MAXVALUE
   CACHE 1;

CREATE SEQUENCE test_id_seq
   START WITH 7000
   INCREMENT BY 1
   NO MINVALUE
   NO MAXVALUE
   CACHE 1;

CREATE SEQUENCE bill_id_sequen
   START WITH 5000
   INCREMENT BY 1
   NO MINVALUE
   NO MAXVALUE
   CACHE 1;

CREATE OR REPLACE FUNCTION assign_patient_id()
RETURNS TRIGGER AS $$
BEGIN
   NEW.PatientID := nextval('patient_id_seq');
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE OR REPLACE FUNCTION assign_doctor_id()
RETURNS TRIGGER AS $$
BEGIN
```

```
        NEW.DoctorID := nextval('doctor_id_seq');
        RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION assign_appointment_id()
RETURNS TRIGGER AS $$
BEGIN
        NEW.AppointmentID := nextval('appointment_id_seq');
        RETURN NEW;
END;
$$ LANGUAGE plpgsql;



CREATE OR REPLACE FUNCTION assign_record_id()
RETURNS TRIGGER AS $$
BEGIN
   NEW.RecordID := nextval('record_id_seq');
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;



CREATE OR REPLACE FUNCTION assign_test_id()
RETURNS TRIGGER AS $$
BEGIN
   NEW.TestID := nextval('test_id_seq');
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION assign_bill_id()
RETURNS TRIGGER AS $$
BEGIN
   NEW.BillID := nextval('bill_id_sequen');
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER assign_bill_id_trigger
BEFORE INSERT ON Billing
FOR EACH ROW
EXECUTE FUNCTION assign_bill_id();
```

```
CREATE TRIGGER assign_test_id_trigger
BEFORE INSERT ON Lab_Tests
FOR EACH ROW
EXECUTE FUNCTION assign_test_id();


CREATE TRIGGER assign_record_id_trigger
BEFORE INSERT ON MedicalRecord
FOR EACH ROW
EXECUTE FUNCTION assign_record_id();


CREATE TRIGGER assign_appointment_id_trigger
BEFORE INSERT ON Appointment
FOR EACH ROW
EXECUTE FUNCTION assign_appointment_id();


CREATE TRIGGER assign_doctor_id
BEFORE INSERT ON Doctor
FOR EACH ROW
EXECUTE PROCEDURE assign_doctor_id();


CREATE TRIGGER assign_patient_id_trigger
BEFORE INSERT ON Patients
FOR EACH ROW
EXECUTE FUNCTION assign_patient_id();
```

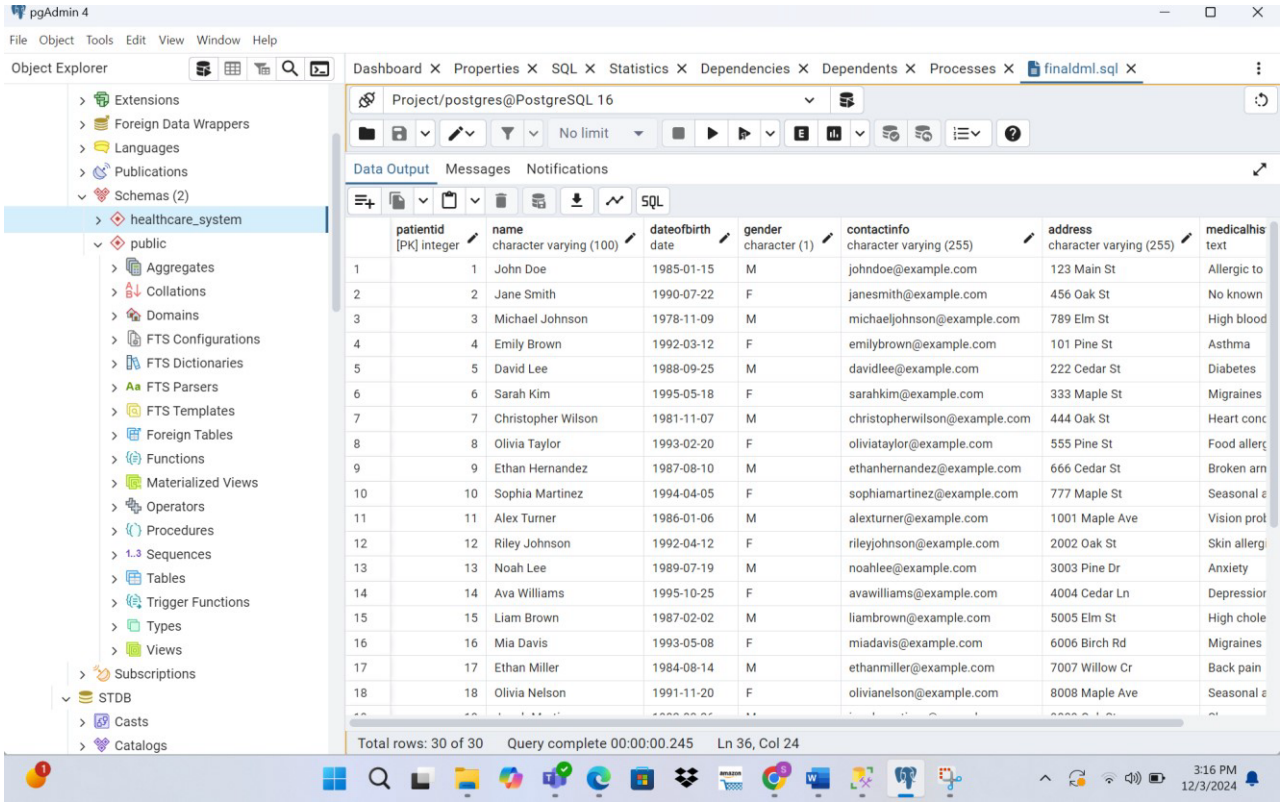
ii. Data Manipulation Language (DML)

a- DML CODE:

--------- Patient table inserts

```
INSERT INTO Patients (Name, DateOfBirth, Gender, ContactInfo, Address, MedicalHistory) VALUES
 ('John Doe', '1985-01-15', 'M', 'johndoe@example.com', '123 Main St', 'Allergic to peanuts'),
 ('Jane Smith', '1990-07-22', 'F', 'janesmith@example.com', '456 Oak St', 'No known allergies'),
('Michael Johnson', '1978-11-09', 'M', 'michaeljohnson@example.com', '789 Elm St', 'High blood
pressure'),
 ('Emily Brown', '1992-03-12', 'F', 'emilybrown@example.com', '101 Pine St', 'Asthma'),
 ('David Lee', '1988-09-25', 'M', 'davidlee@example.com', '222 Cedar St', 'Diabetes'),
 ('Sarah Kim', '1995-05-18', 'F', 'sarahkim@example.com', '333 Maple St', 'Migraines'),
 ('Christopher Wilson', '1981-11-07', 'M', 'christopherwilson@example.com', '444 Oak St', 'Heart
condition'),
 ('Olivia Taylor', '1993-02-20', 'F', 'oliviataylor@example.com', '555 Pine St', 'Food allergies'),
```

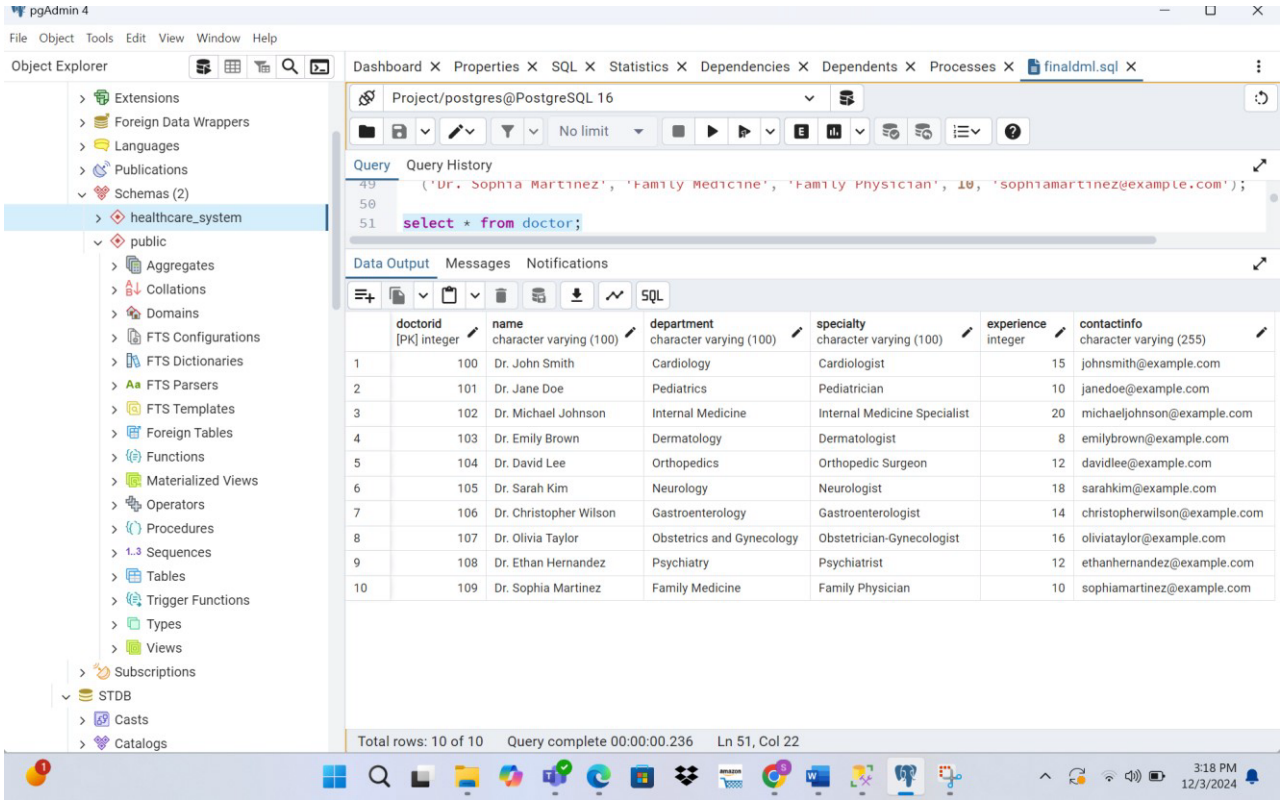
('Ethan Hernandez', '1987-08-10', 'M', 'ethanhernandez@example.com', '666 Cedar St', 'Broken arm (past injury)'),
('Sophia Martinez', '1994-04-05', 'F', 'sophiamartinez@example.com', '777 Maple St', 'Seasonal allergies'),
('Alex Turner', '1986-01-06', 'M', 'alexturner@example.com', '1001 Maple Ave', 'Vision problems'),
('Riley Johnson', '1992-04-12', 'F', 'rileyjohnson@example.com', '2002 Oak St', 'Skin allergies'),
('Noah Lee', '1989-07-19', 'M', 'noahlee@example.com', '3003 Pine Dr', 'Anxiety'),
('Ava Williams', '1995-10-25', 'F', 'avawilliams@example.com', '4004 Cedar Ln', 'Depression'),
('Liam Brown', '1987-02-02', 'M', 'liambrown@example.com', '5005 Elm St', 'High cholesterol'),
('Mia Davis', '1993-05-08', 'F', 'miadavis@example.com', '6006 Birch Rd', 'Migraines'),
('Ethan Miller', '1984-08-14', 'M', 'ethanmiller@example.com', '7007 Willow Cr', 'Back pain'),
('Olivia Nelson', '1991-11-20', 'F', 'olivianelson@example.com', '8008 Maple Ave', 'Seasonal allergies'),
('Jacob Martinez', '1982-03-26', 'M', 'jacobmartinez@example.com', '9009 Oak St', 'Sleep apnea'),
('Sophia Taylor', '1996-06-01', 'F', 'sophiataylor@example.com', '1010 Pine Dr', 'No known issues'),
('Benjamin Carter', '1988-07-12', 'M', 'benjamincarter@example.com', '1111 Elm St', 'Food allergies'),
('Charlotte Lee', '1994-10-23', 'F', 'charlottelee@example.com', '2222 Pine Ave', 'Anxiety'),
('Daniel Johnson', '1985-01-29', 'M', 'danieljohnson@example.com', '3333 Oak Ln', 'High blood pressure'),
('Eleanor Davis', '1991-04-04', 'F', 'eleanordavis@example.com', '4444 Maple Rd', 'Depression'),
('Finnley Wilson', '1987-07-10', 'M', 'finnleywilson@example.com', '5555 Cedar Dr', 'Migraines'),
('Grace Thomas', '1993-10-16', 'F', 'gracethomas@example.com', '6666 Birch Ct', 'Asthma'),
('Henry Martinez', '1989-01-22', 'M', 'henrymartinez@example.com', '7777 Willow Ln', 'Vision problems'),
('Isabella Harris', '1995-04-28', 'F', 'isabellaharris@example.com', '8888 Elm St', 'Skin allergies'),
('Jackson Smith', '1986-07-03', 'M', 'jacksonsmith@example.com', '9999 Pine Ave', 'Sleep apnea'),
('Katherine Miller', '1992-10-09', 'F', 'katherinemiller@example.com', '1010 Oak Ln', 'No known issues');
select * from patients;



------Insert records into Doctor table

```
INSERT INTO Doctor (Name, Department, Specialty, Experience, ContactInfo) VALUES
  ('Dr. John Smith', 'Cardiology', 'Cardiologist', 15, 'johnsmith@example.com'),
  ('Dr. Jane Doe', 'Pediatrics', 'Pediatrician', 10, 'janedoe@example.com'),
  ('Dr. Michael Johnson', 'Internal Medicine', 'Internal Medicine Specialist', 20,
'michaeljohnson@example.com'),
  ('Dr. Emily Brown', 'Dermatology', 'Dermatologist', 8, 'emilybrown@example.com'),
  ('Dr. David Lee', 'Orthopedics', 'Orthopedic Surgeon', 12, 'davidlee@example.com'),
  ('Dr. Sarah Kim', 'Neurology', 'Neurologist', 18, 'sarahkim@example.com'),
  ('Dr. Christopher Wilson', 'Gastroenterology', 'Gastroenterologist', 14,
'christopherwilson@example.com'),
  ('Dr. Olivia Taylor', 'Obstetrics and Gynecology', 'Obstetrician-Gynecologist', 16,
'oliviataylor@example.com'),
  ('Dr. Ethan Hernandez', 'Psychiatry', 'Psychiatrist', 12, 'ethanhernandez@example.com'),
  ('Dr. Sophia Martinez', 'Family Medicine', 'Family Physician', 10, 'sophiamartinez@example.com');
```

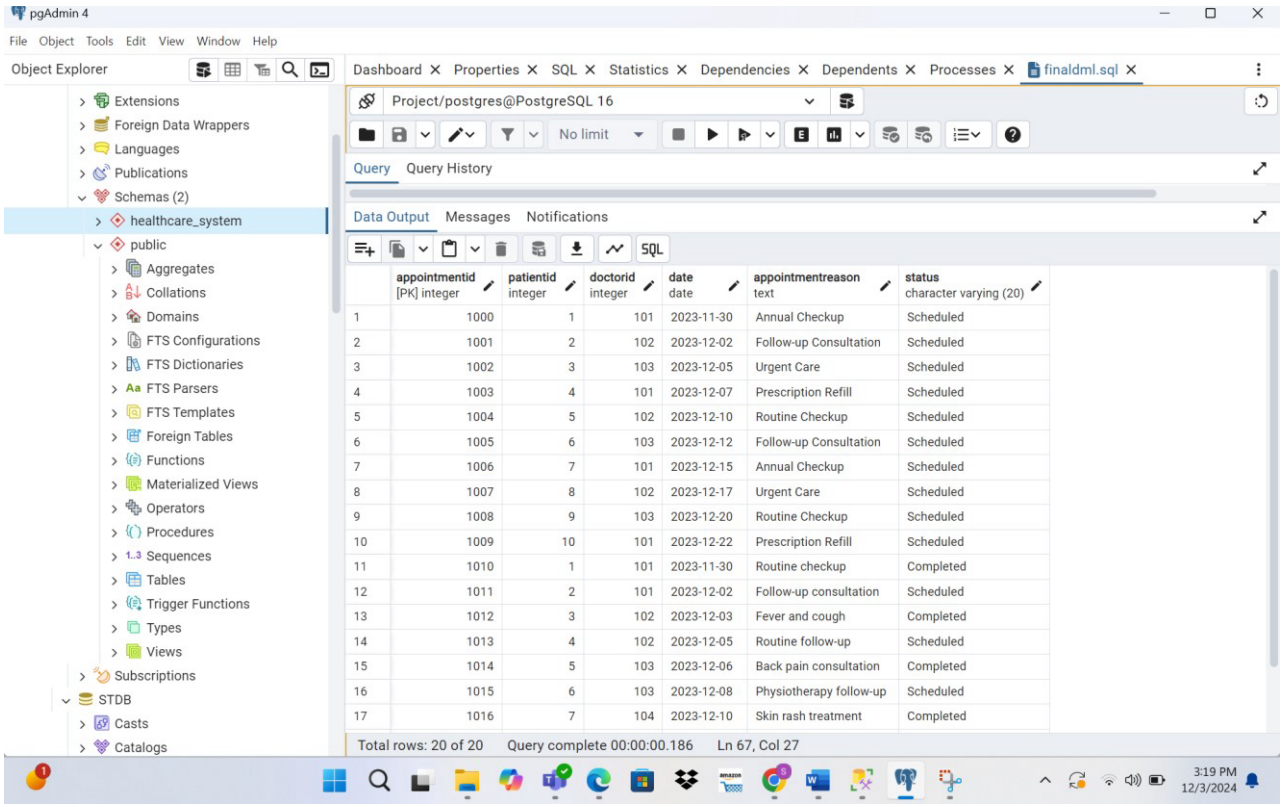
```
select * from doctor;
```



doctorid [PK] integer	name character varying (100)	department character varying (100)	specialty character varying (100)	experience integer	contactinfo character varying (255)
1	100	Dr. John Smith	Cardiology	Cardiologist	15	johnsmith@example.com
2	101	Dr. Jane Doe	Pediatrics	Pediatrician	10	janedoe@example.com
3	102	Dr. Michael Johnson	Internal Medicine	Internal Medicine Specialist	20	michaeljohnson@example.com
4	103	Dr. Emily Brown	Dermatology	Dermatologist	8	emilybrown@example.com
5	104	Dr. David Lee	Orthopedics	Orthopedic Surgeon	12	davidlee@example.com
6	105	Dr. Sarah Kim	Neurology	Neurologist	18	sarahkim@example.com
7	106	Dr. Christopher Wilson	Gastroenterology	Gastroenterologist	14	christopherwilson@example.com
8	107	Dr. Olivia Taylor	Obstetrics and Gynecology	Obstetrician-Gynecologist	16	oliviataylor@example.com
9	108	Dr. Ethan Hernandez	Psychiatry	Psychiatrist	12	ethanhernandez@example.com
10	109	Dr. Sophia Martinez	Family Medicine	Family Physician	10	sophiamartinez@example.com

------- Insert records into Appointment Table

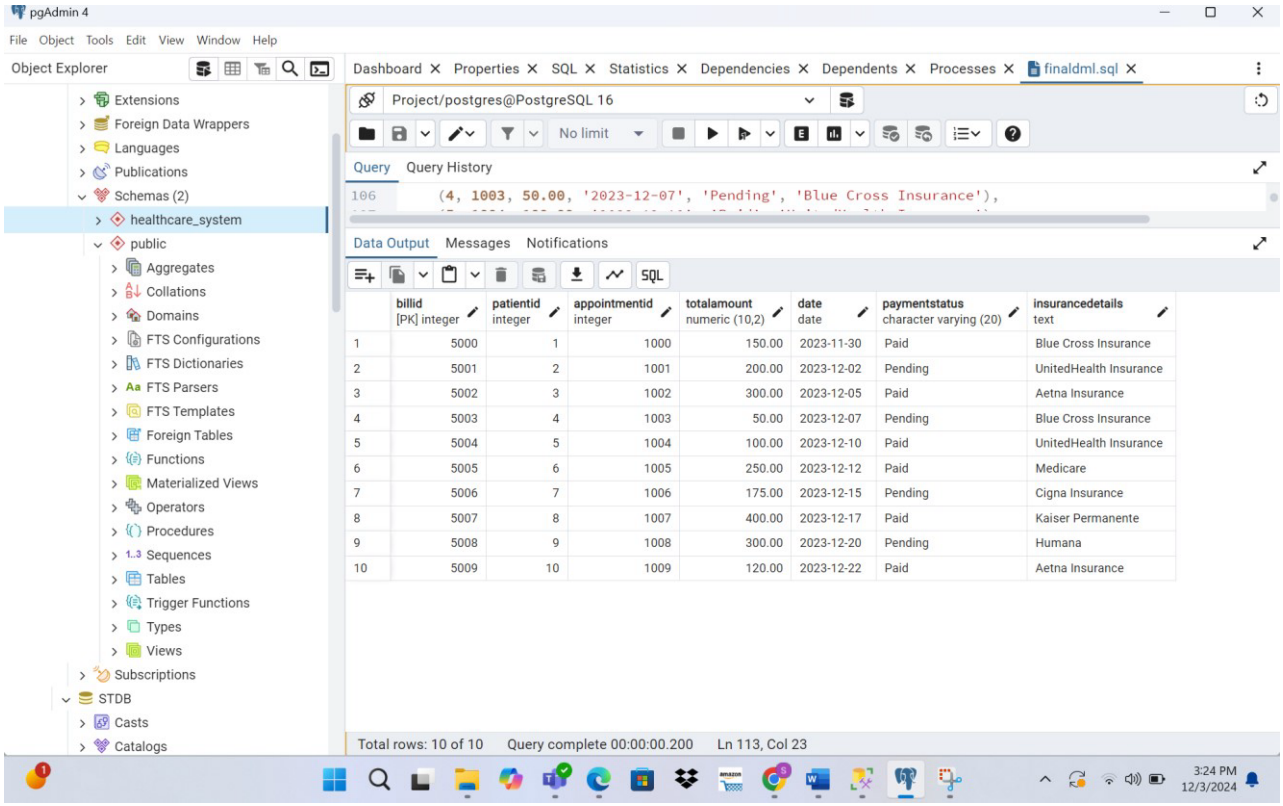
```
INSERT INTO Appointment (PatientID, DoctorID, Date, AppointmentReason, Status)
VALUES
        (1, 101, '2023-11-30', 'Annual Checkup', 'Scheduled'),
        (2, 102, '2023-12-02', 'Follow-up Consultation', 'Scheduled'),
        (3, 103, '2023-12-05', 'Urgent Care', 'Scheduled'),
        (4, 101, '2023-12-07', 'Prescription Refill', 'Scheduled'),
        (5, 102, '2023-12-10', 'Routine Checkup', 'Scheduled'),
        (6, 103, '2023-12-12', 'Follow-up Consultation', 'Scheduled'),
        (7, 101, '2023-12-15', 'Annual Checkup', 'Scheduled'),
        (8, 102, '2023-12-17', 'Urgent Care', 'Scheduled'),
        (9, 103, '2023-12-20', 'Routine Checkup', 'Scheduled'),
        (10, 101, '2023-12-22', 'Prescription Refill', 'Scheduled');

select * from Appointment;
```



---------- Insert records into MedicalRecord

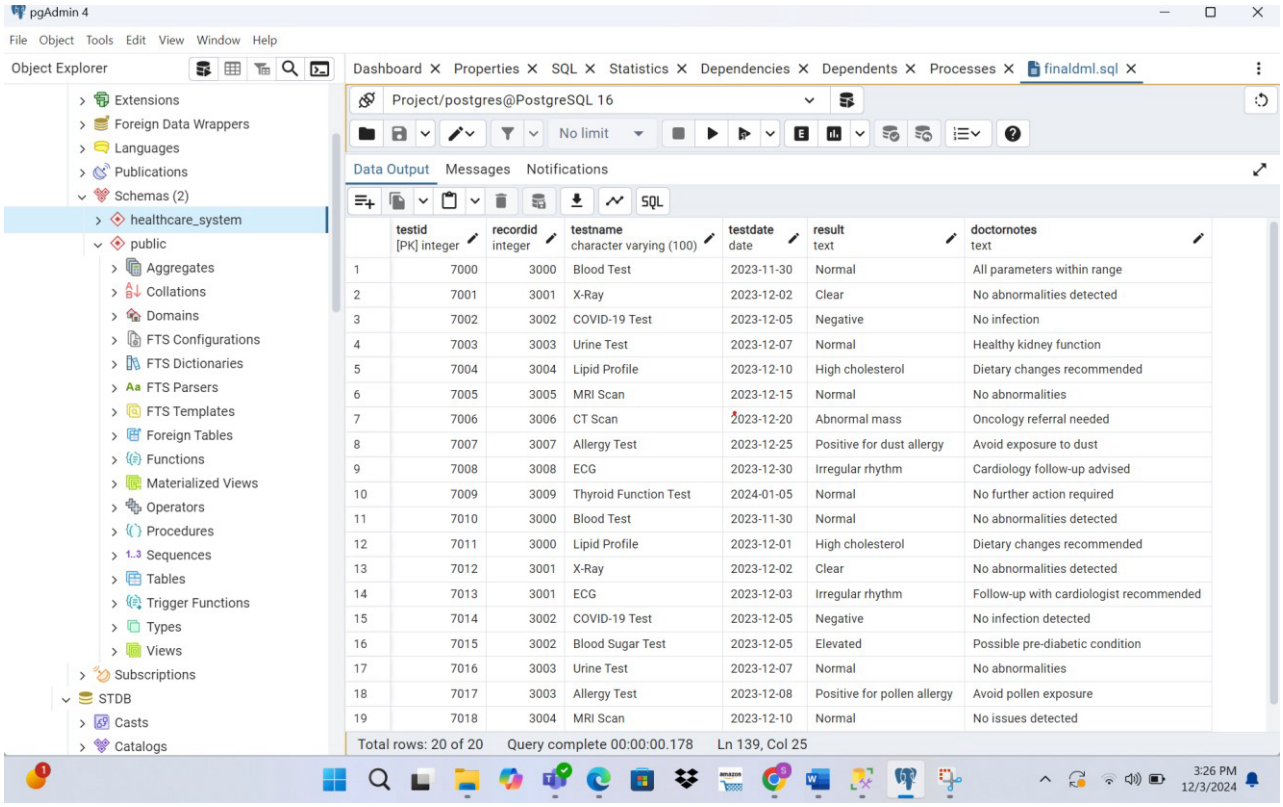
```
INSERT INTO MedicalRecord (PatientID, AppointmentID, DoctorID, Notes, Treatment) VALUES
    (1, 1000, 101, 'Annual checkup performed. No issues.', 'Routine physical exam'),
    (2, 1001, 102, 'Follow-up consultation for ongoing condition.', 'Medication adjusted'),
    (3, 1002, 103, 'Presented with fever and cough.', 'Prescribed antibiotics'),
    (4, 1003, 101, 'Requested refill for regular medication.', 'Medication refilled'),
    (5, 1004, 102, 'Routine checkup completed.', 'Blood tests ordered'),
    (6, 1005, 103, 'Patient reported back pain.', 'Prescribed physiotherapy'),
```

```
    (7, 1006, 104, 'Skin rash diagnosed.', 'Prescribed ointment'),
    (8, 1007, 105, 'Complained of frequent headaches.', 'MRI scan recommended'),
    (9, 1008, 106, 'Routine dental checkup.', 'Teeth cleaned'),
    (10, 1009, 107, 'Vision issues reported.', 'Prescribed glasses.');


select * from MedicalRecord;


-------insert records into Lab_Tests

INSERT INTO Lab_Tests (RecordID, TestName, TestDate, Result, DoctorNotes) VALUES
    (3000, 'Blood Test', '2023-11-30', 'Normal', 'All parameters within range'),
    (3001, 'X-Ray', '2023-12-02', 'Clear', 'No abnormalities detected'),
    (3002, 'COVID-19 Test', '2023-12-05', 'Negative', 'No infection'),
    (3003, 'Urine Test', '2023-12-07', 'Normal', 'Healthy kidney function'),
    (3004, 'Lipid Profile', '2023-12-10', 'High cholesterol', 'Dietary changes recommended'),
    (3005, 'MRI Scan', '2023-12-15', 'Normal', 'No abnormalities'),
    (3006, 'CT Scan', '2023-12-20', 'Abnormal mass', 'Oncology referral needed'),
    (3007, 'Allergy Test', '2023-12-25', 'Positive for dust allergy', 'Avoid exposure to dust'),
    (3008, 'ECG', '2023-12-30', 'Irregular rhythm', 'Cardiology follow-up advised'),
    (3009, 'Thyroid Function Test', '2024-01-05', 'Normal', 'No further action required');

select * from Lab_Tests;


---------- insert records into Billing Table

INSERT INTO Billing (PatientID, AppointmentID, TotalAmount, Date, PaymentStatus, InsuranceDetails)
VALUES
    (1, 1000, 150.00, '2023-11-30', 'Paid', 'Blue Cross Insurance'),
    (2, 1001, 200.00, '2023-12-02', 'Pending', 'UnitedHealth Insurance'),
    (3, 1002, 300.00, '2023-12-05', 'Paid', 'Aetna Insurance'),
    (4, 1003, 50.00, '2023-12-07', 'Pending', 'Blue Cross Insurance'),
    (5, 1004, 100.00, '2023-12-10', 'Paid', 'UnitedHealth Insurance'),
    (6, 1005, 250.00, '2023-12-12', 'Paid', 'Medicare'),
    (7, 1006, 175.00, '2023-12-15', 'Pending', 'Cigna Insurance'),
    (8, 1007, 400.00, '2023-12-17', 'Paid', 'Kaiser Permanente'),
    (9, 1008, 300.00, '2023-12-20', 'Pending', 'Humana'),
    (10, 1009, 120.00, '2023-12-22', 'Paid', 'Aetna Insurance');

SELECT * FROM Billing;
```



-------Insert records into Lab_Test Table

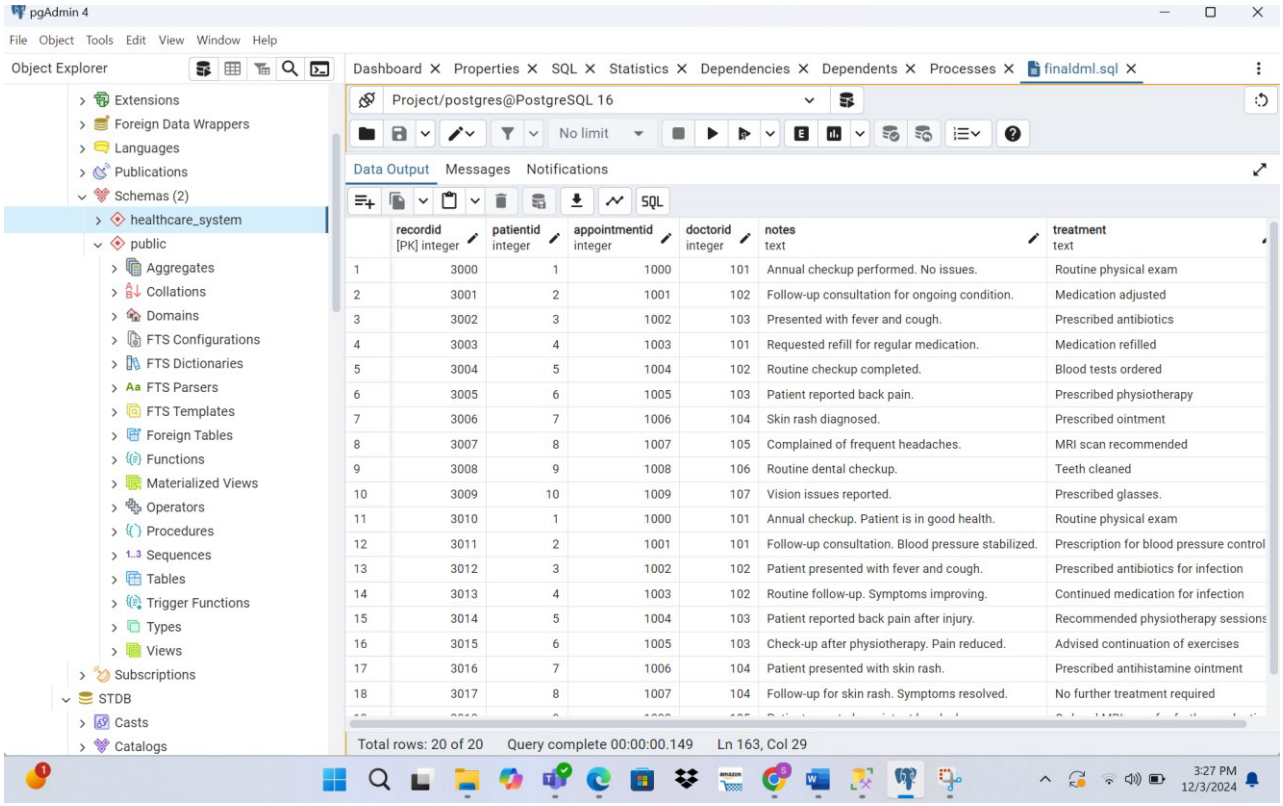
```
INSERT INTO Lab_Tests (RecordID, TestName, TestDate, Result, DoctorNotes)
VALUES
   -- Lab tests for MedicalRecord with RecordID = 3000
   (3000, 'Blood Test', '2023-11-30', 'Normal', 'No abnormalities detected'),
   (3000, 'Lipid Profile', '2023-12-01', 'High cholesterol', 'Dietary changes recommended'),

   -- Lab tests for MedicalRecord with RecordID = 3001
   (3001, 'X-Ray', '2023-12-02', 'Clear', 'No abnormalities detected'),
   (3001, 'ECG', '2023-12-03', 'Irregular rhythm', 'Follow-up with cardiologist recommended'),

   -- Lab tests for MedicalRecord with RecordID = 3002
   (3002, 'COVID-19 Test', '2023-12-05', 'Negative', 'No infection detected'),
   (3002, 'Blood Sugar Test', '2023-12-05', 'Elevated', 'Possible pre-diabetic condition'),

   -- Lab tests for MedicalRecord with RecordID = 3003
   (3003, 'Urine Test', '2023-12-07', 'Normal', 'No abnormalities'),
   (3003, 'Allergy Test', '2023-12-08', 'Positive for pollen allergy', 'Avoid pollen exposure'),

   -- Lab tests for MedicalRecord with RecordID = 3004
   (3004, 'MRI Scan', '2023-12-10', 'Normal', 'No issues detected'),
   (3004, 'Thyroid Function Test', '2023-12-11', 'Normal', 'No further action required');

select * from lab_tests;
```



	testid [PK] integer	recordid integer	testname character varying (100)	testdate date	result text	doctornotes text
1	7000	3000	Blood Test	2023-11-30	Normal	All parameters within range
2	7001	3001	X-Ray	2023-12-02	Clear	No abnormalities detected
3	7002	3002	COVID-19 Test	2023-12-05	Negative	No infection
4	7003	3003	Urine Test	2023-12-07	Normal	Healthy kidney function
5	7004	3004	Lipid Profile	2023-12-10	High cholesterol	Dietary changes recommended
6	7005	3005	MRI Scan	2023-12-15	Normal	No abnormalities
7	7006	3006	CT Scan	2023-12-20	Abnormal mass	Oncology referral needed
8	7007	3007	Allergy Test	2023-12-25	Positive for dust allergy	Avoid exposure to dust
9	7008	3008	ECG	2023-12-30	Irregular rhythm	Cardiology follow-up advised
10	7009	3009	Thyroid Function Test	2024-01-05	Normal	No further action required
11	7010	3000	Blood Test	2023-11-30	Normal	No abnormalities detected
12	7011	3000	Lipid Profile	2023-12-01	High cholesterol	Dietary changes recommended
13	7012	3001	X-Ray	2023-12-02	Clear	No abnormalities detected
14	7013	3001	ECG	2023-12-03	Irregular rhythm	Follow-up with cardiologist recommended
15	7014	3002	COVID-19 Test	2023-12-05	Negative	No infection detected
16	7015	3002	Blood Sugar Test	2023-12-05	Elevated	Possible pre-diabetic condition
17	7016	3003	Urine Test	2023-12-07	Normal	No abnormalities
18	7017	3003	Allergy Test	2023-12-08	Positive for pollen allergy	Avoid pollen exposure
19	7018	3004	MRI Scan	2023-12-10	Normal	No issues detected

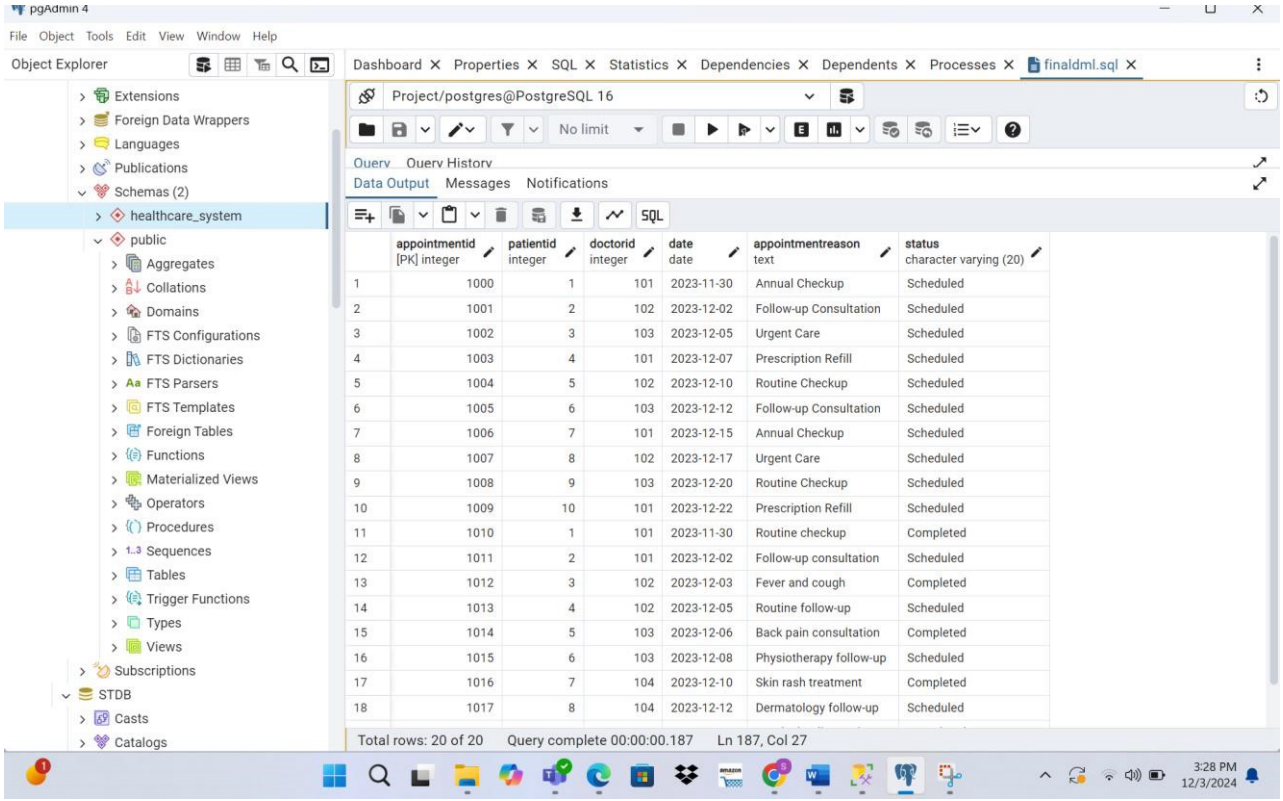
--------Insert records into MedicalRecord Table

```
INSERT INTO MedicalRecord (PatientID, AppointmentID, DoctorID, Notes, Treatment) VALUES
   -- Records associated with DoctorID = 101
   (1, 1000, 101, 'Annual checkup. Patient is in good health.', 'Routine physical exam'),
   (2, 1001, 101, 'Follow-up consultation. Blood pressure stabilized.', 'Prescription for blood pressure control'),

   -- Records associated with DoctorID = 102
   (3, 1002, 102, 'Patient presented with fever and cough.', 'Prescribed antibiotics for infection'),
   (4, 1003, 102, 'Routine follow-up. Symptoms improving.', 'Continued medication for infection'),

   -- Records associated with DoctorID = 103
   (5, 1004, 103, 'Patient reported back pain after injury.', 'Recommended physiotherapy sessions'),
   (6, 1005, 103, 'Check-up after physiotherapy. Pain reduced.', 'Advised continuation of exercises'),

   -- Records associated with DoctorID = 104
   (7, 1006, 104, 'Patient presented with skin rash.', 'Prescribed antihistamine ointment'),
   (8, 1007, 104, 'Follow-up for skin rash. Symptoms resolved.', 'No further treatment required'),

   -- Records associated with DoctorID = 105
   (9, 1008, 105, 'Patient reported persistent headaches.', 'Ordered MRI scan for further evaluation'),
   (10, 1009, 105, 'MRI results normal. Advised stress management.', 'No medical intervention required');

select * from MedicalRecord;
```



------Insert records into Appointment Table

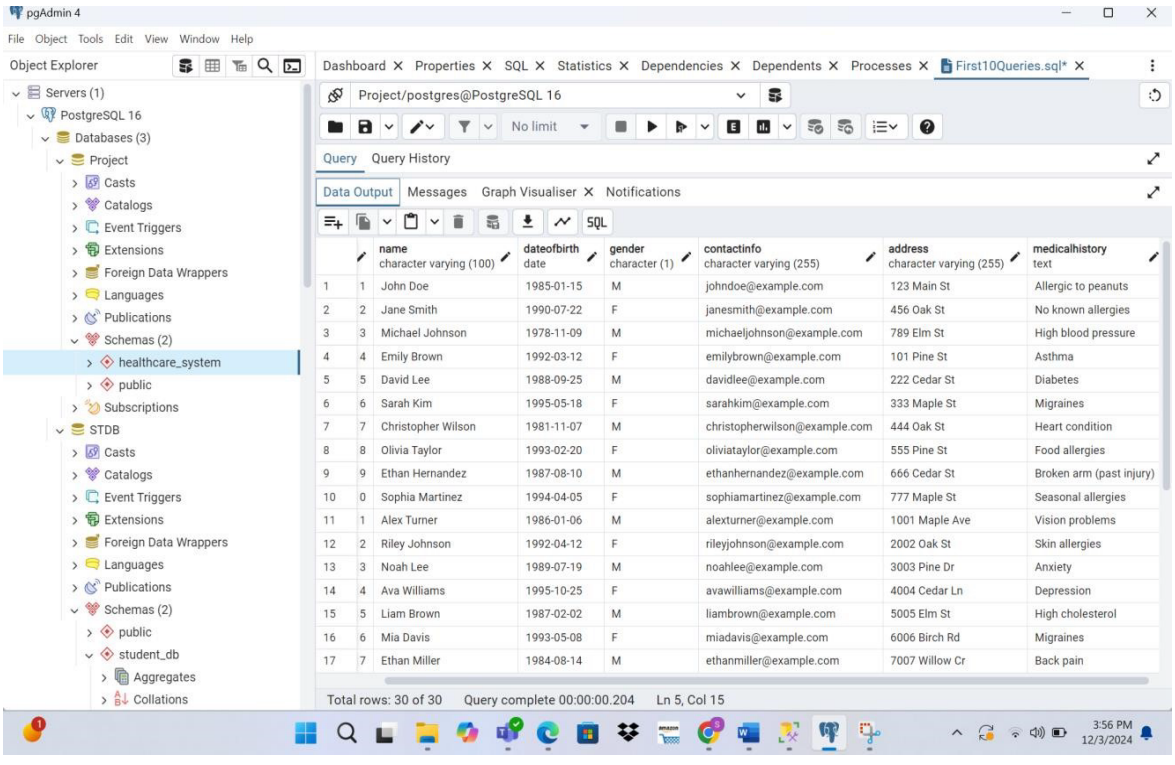
```
INSERT INTO Appointment (PatientID, DoctorID, Date, AppointmentReason, Status) VALUES
   -- Appointments for DoctorID = 101
   (1, 101, '2023-11-30', 'Routine checkup', 'Completed'),
   (2, 101, '2023-12-02', 'Follow-up consultation', 'Scheduled'),

   -- Appointments for DoctorID = 102
   (3, 102, '2023-12-03', 'Fever and cough', 'Completed'),
   (4, 102, '2023-12-05', 'Routine follow-up', 'Scheduled'),

   -- Appointments for DoctorID = 103
   (5, 103, '2023-12-06', 'Back pain consultation', 'Completed'),
   (6, 103, '2023-12-08', 'Physiotherapy follow-up', 'Scheduled'),

   -- Appointments for DoctorID = 104
   (7, 104, '2023-12-10', 'Skin rash treatment', 'Completed'),
   (8, 104, '2023-12-12', 'Dermatology follow-up', 'Scheduled'),

   -- Appointments for DoctorID = 105
   (9, 105, '2023-12-14', 'Headache diagnosis', 'Completed'),
   (10, 105, '2023-12-16', 'MRI scan review', 'Scheduled');

select * from Appointment;
```



b) SQL Query code and Output:

➢ **Query 1: Select all columns and all rows from one table.**

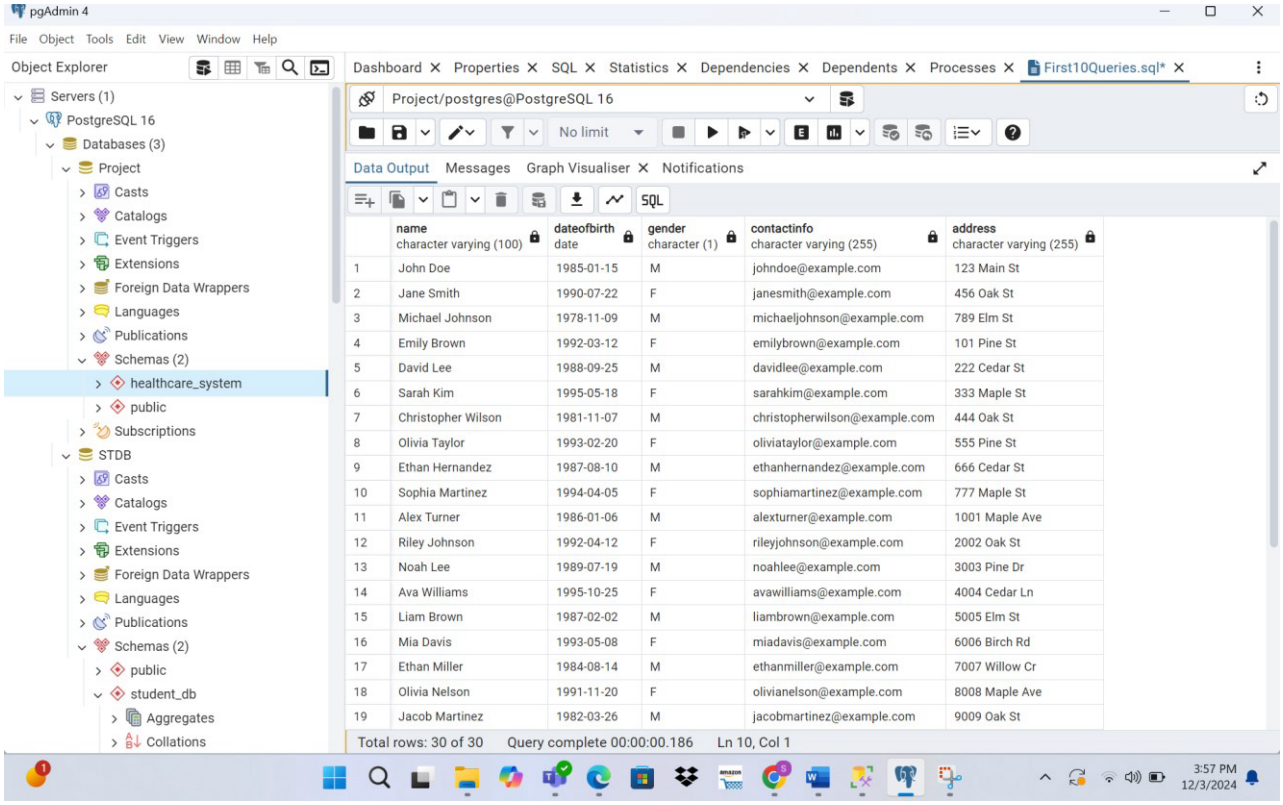
```
SELECT *
FROM Patients;
```



➢

Query 2: Select five columns and all rows from one table.

SELECT Name, DateOfBirth, Gender, ContactInfo, Address
FROM Patients;



➢

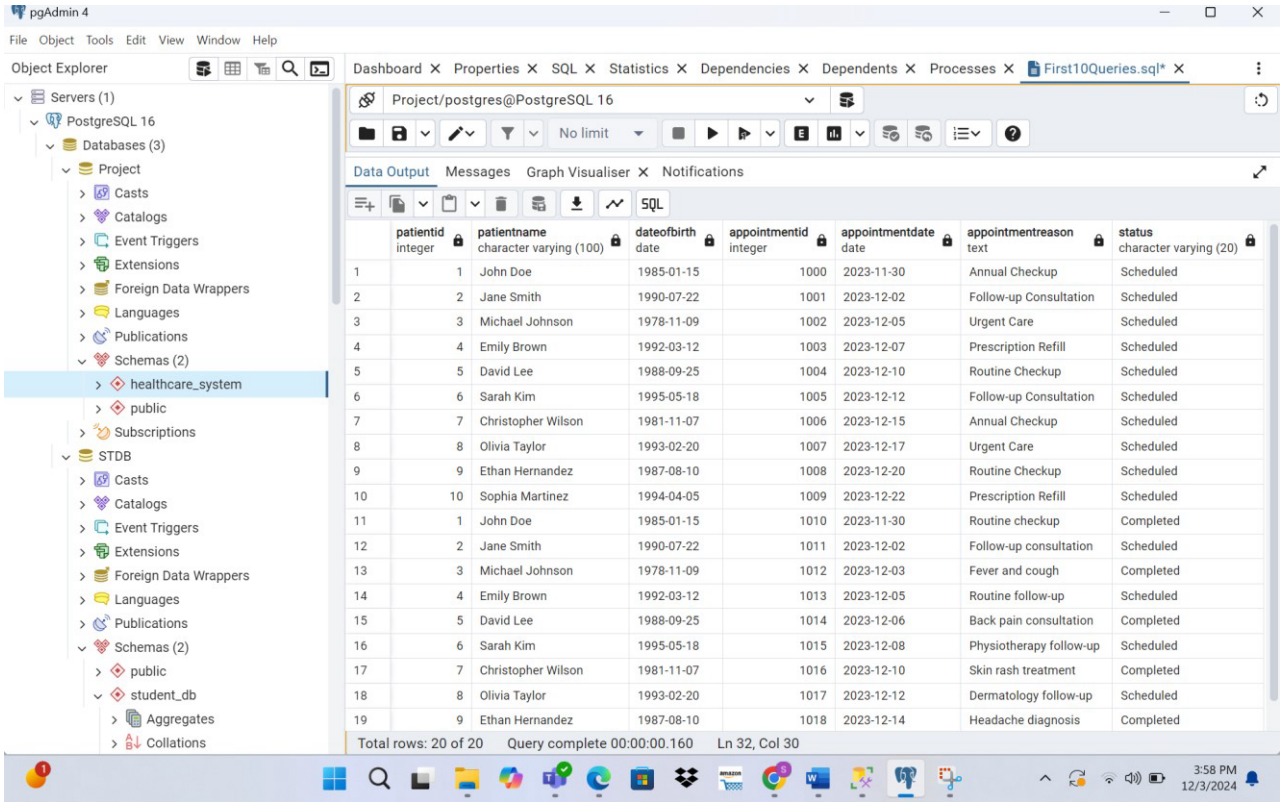
Query 3: Select all columns from all rows from one view.

-----creating a view:

```
CREATE OR REPLACE VIEW PatientAppointmentsView AS
SELECT
   p.PatientID,
   p.Name AS PatientName,
   p.DateOfBirth,
   a.AppointmentID,
   a.Date AS AppointmentDate,
   a.AppointmentReason,
   a.Status
FROM
   Patients p
JOIN
   Appointment a ON p.PatientID = a.PatientID;
```

----------displaying it

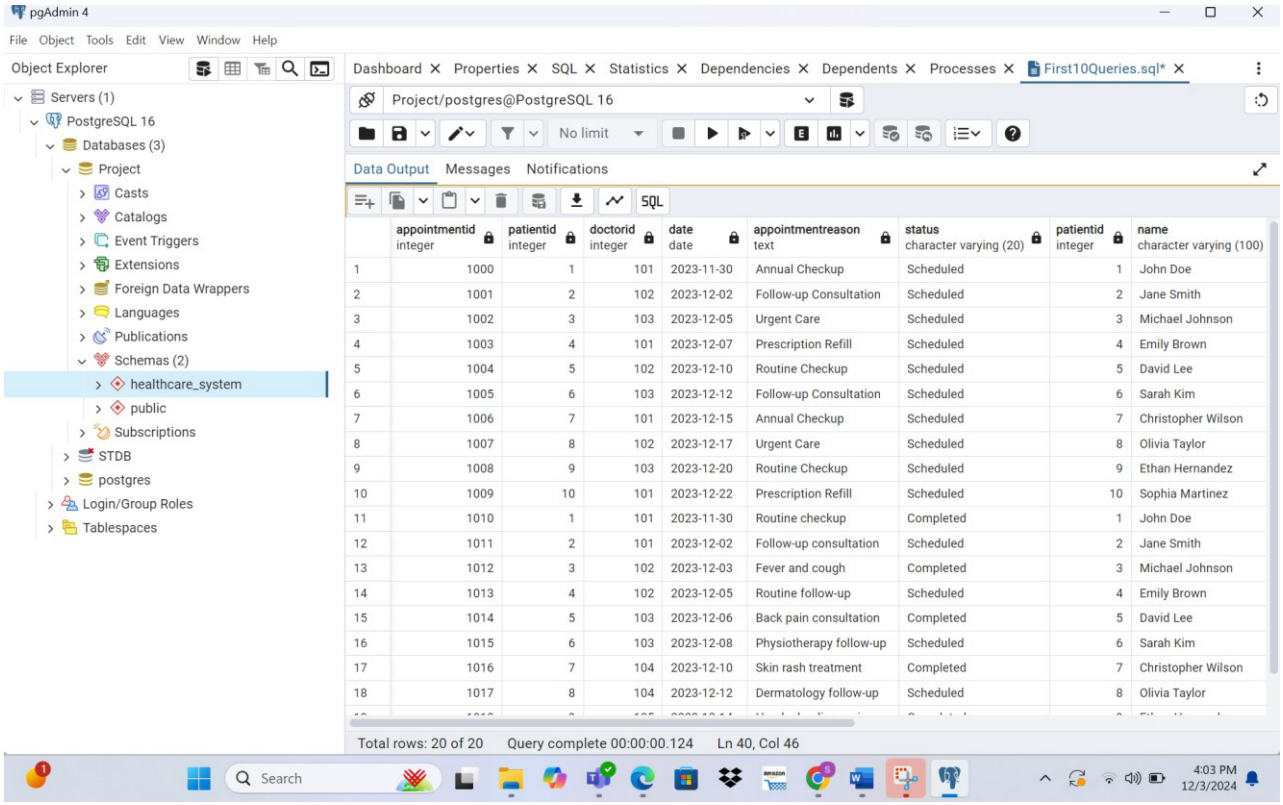
```
SELECT *
FROM PatientAppointmentsView;
```

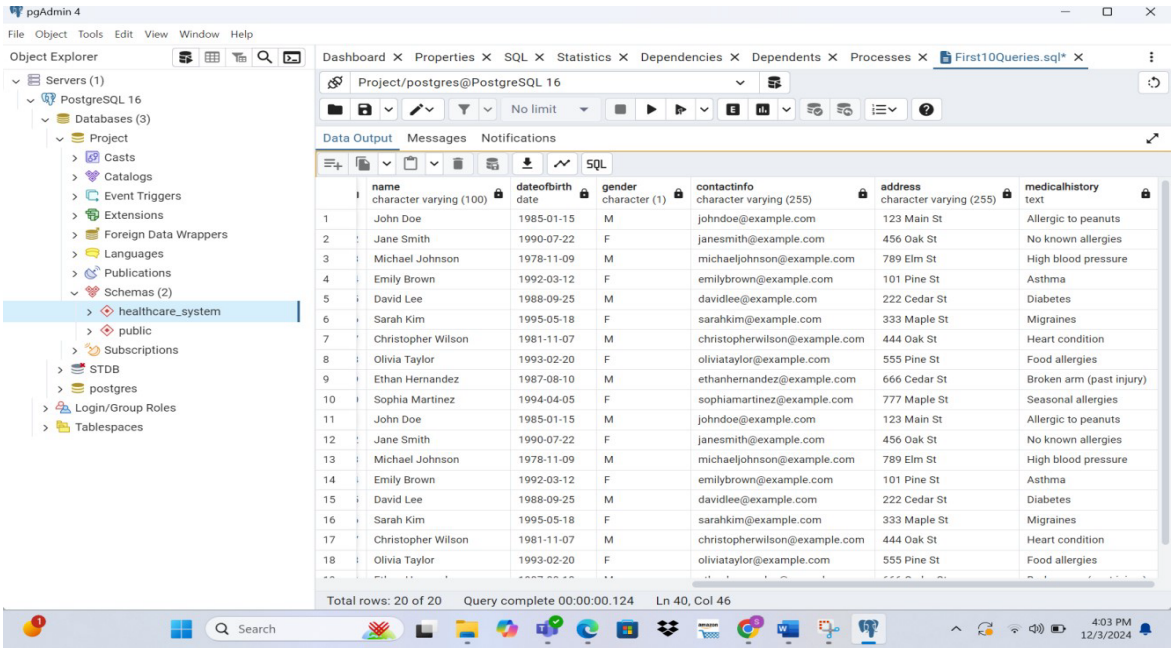


➢

Query 4: Using a join on 2 tables, select all columns and all rows from the tables without the use of a Cartesian product.

```
SELECT *
FROM Appointment a
JOIN Patients p ON a.PatientID = p.PatientID;
```

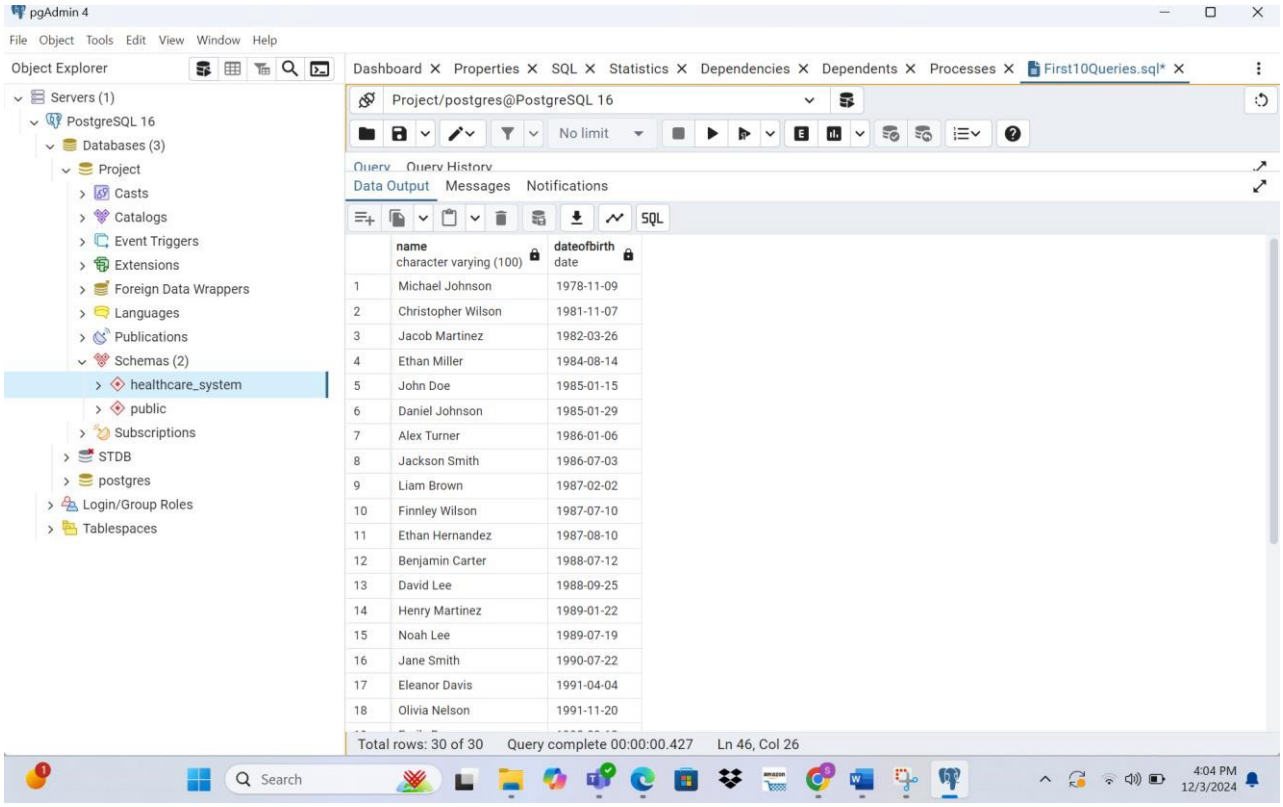




➢

Query 5: Select and order data retrieved from one table.

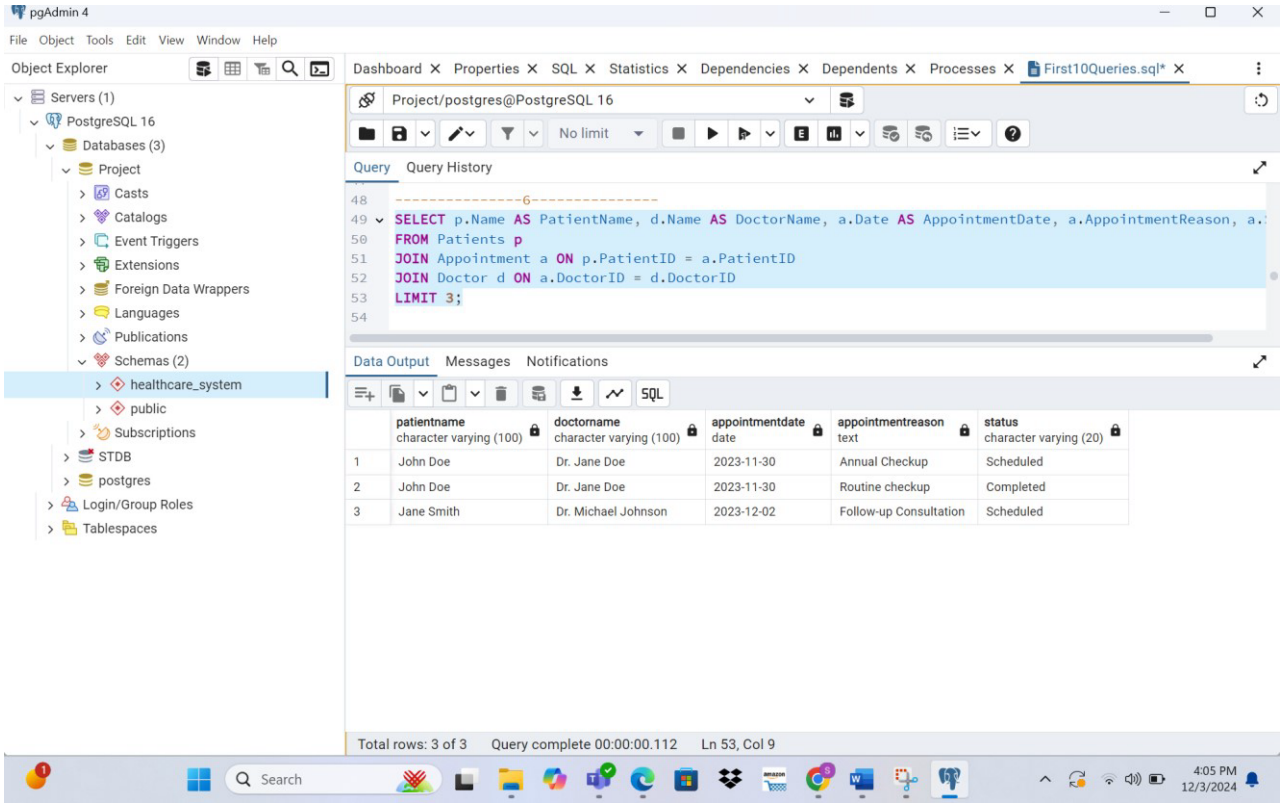
```
SELECT Name, DateOfBirth
FROM Patients
ORDER BY DateOfBirth ASC;
```



➢

Query 6: Using a join on 3 tables, select 5 columns from the 3 tables. Use syntax that would limit the output to 3 rows.

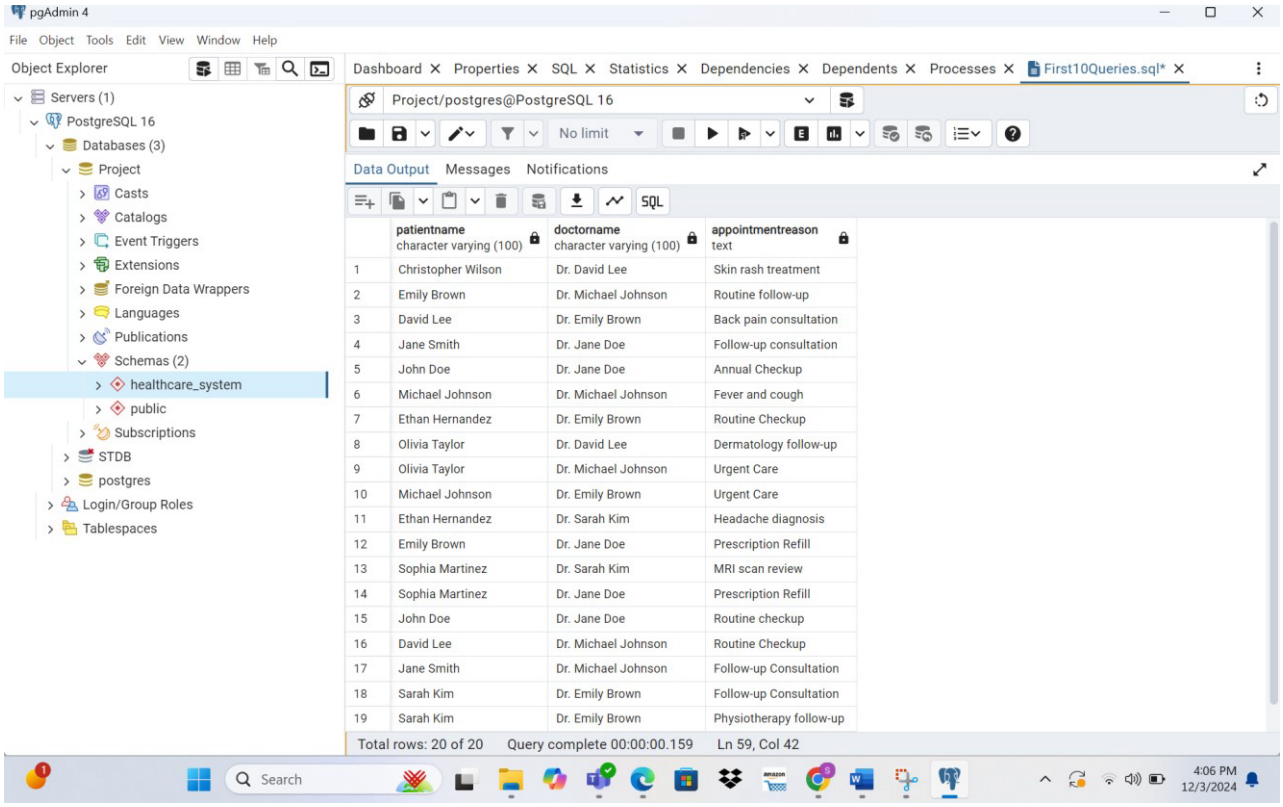
```
SELECT p.Name AS PatientName, d.Name AS DoctorName, a.Date AS AppointmentDate, a.AppointmentReason, a.Status
FROM Patients p
JOIN Appointment a ON p.PatientID = a.PatientID
JOIN Doctor d ON a.DoctorID = d.DoctorID
LIMIT 3;
```



➢

Query 7: Select distinct rows using joins on 3 tables.

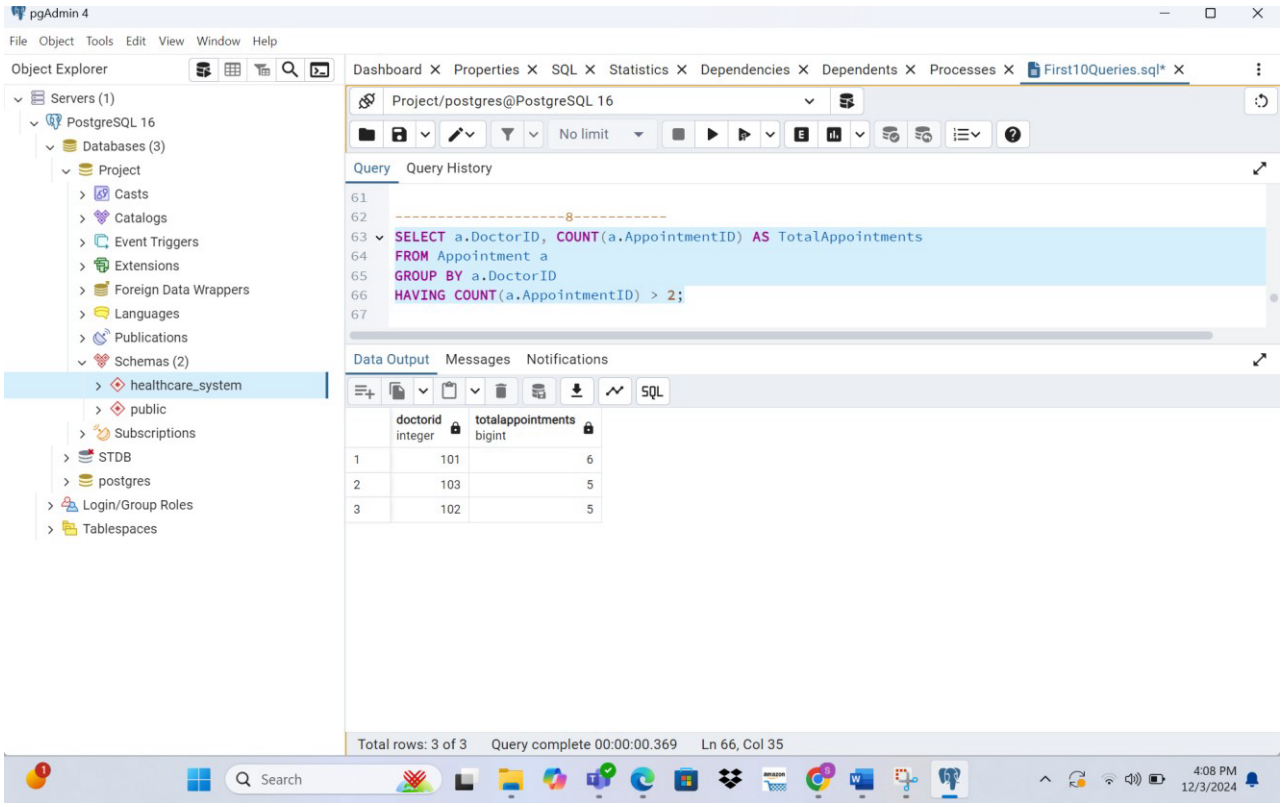
```
SELECT DISTINCT p.Name AS PatientName, d.Name AS DoctorName, a.AppointmentReason
FROM Patients p
JOIN Appointment a ON p.PatientID = a.PatientID
JOIN Doctor d ON a.DoctorID = d.DoctorID;
```



➢

Query 8: Use GROUP BY and HAVING in a select statement using one or more tables.

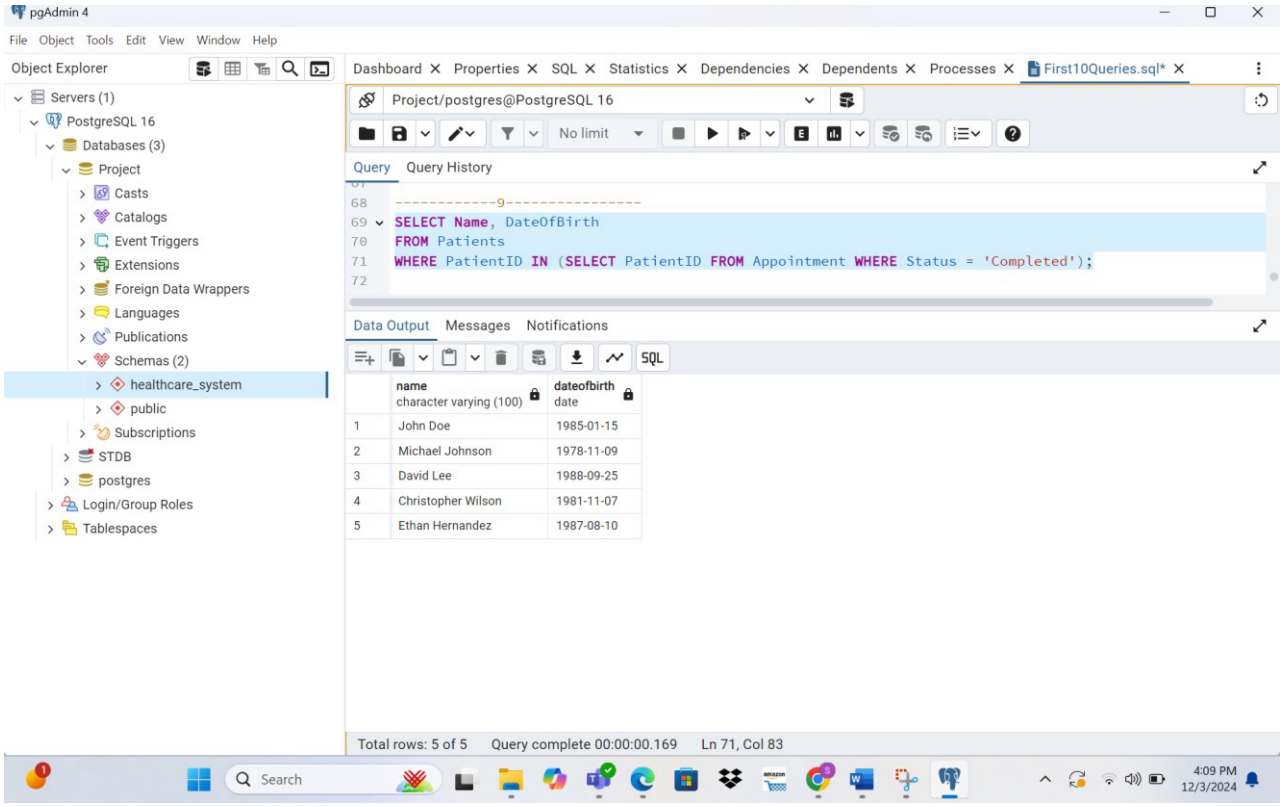
```
SELECT a.DoctorID, COUNT(a.AppointmentID) AS TotalAppointments
FROM Appointment a
GROUP BY a.DoctorID
HAVING COUNT(a.AppointmentID) > 2;
```



➢

Query 9: Use IN clause to select data from one or more tables

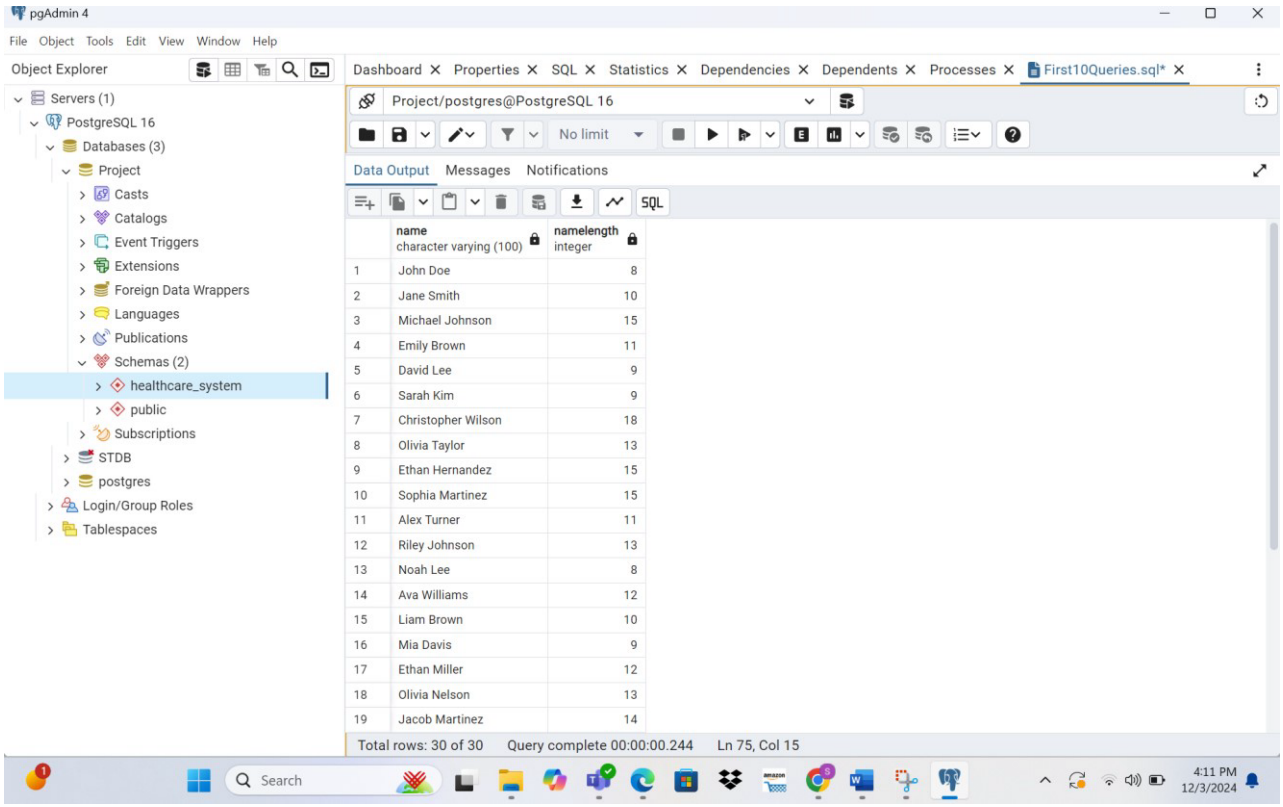
SELECT Name, DateOfBirth
FROM Patients
WHERE PatientID IN (SELECT PatientID FROM Appointment WHERE Status = 'Completed');



➢

Query 10: Select length of one column from one table (use LENGTH function).

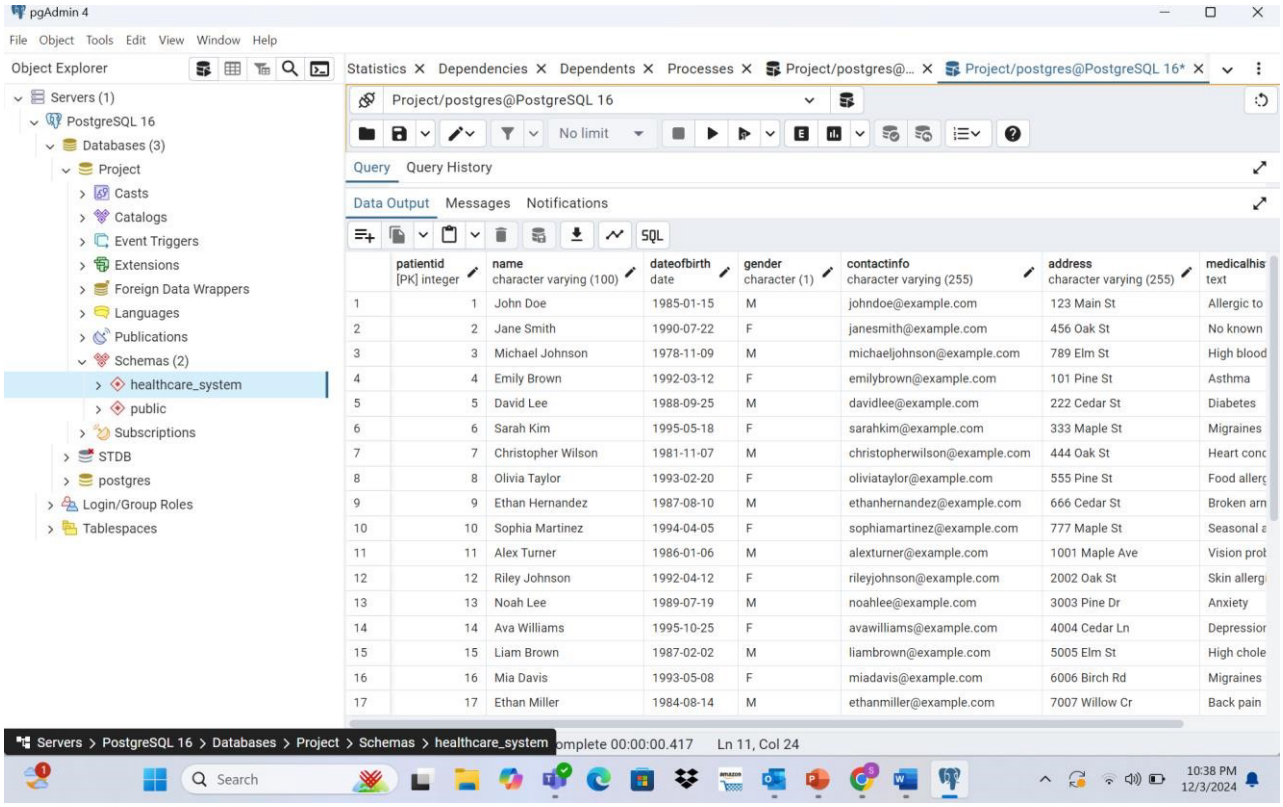
```
SELECT Name, LENGTH(Name) AS NameLength
FROM Patients;
```



➢

Query 11: Delete one record from one table. Use select statements to demonstrate the table contents before and after the DELETE statement. Make sure you use ROLLBACK afterwards so that the data will not be physically removed.

```
SELECT * FROM patients;

BEGIN;
DELETE FROM patients
WHERE patientid = 23;
SELECT * FROM patients;

ROLLBACK;
SELECT * FROM patients;
```

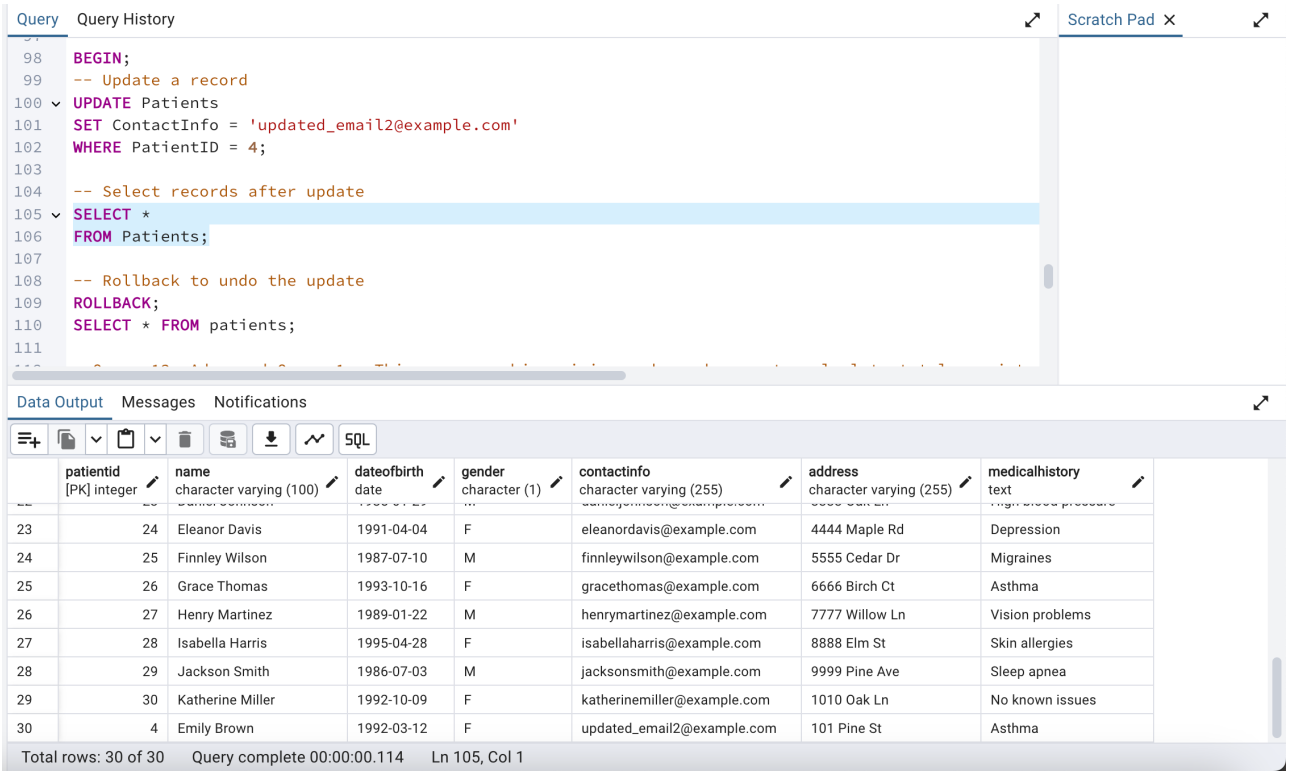


➢

Query 12: Update one record from one table. Use select statements to demonstrate the table contents before and after the UPDATE statement. Make sure you use ROLLBACK afterwards

--Select records before update

```
SELECT *
FROM Patients;
BEGIN;
-- Update a record
UPDATE Patients
SET ContactInfo = 'updated_email2@example.com'
WHERE PatientID = 4;
-- Select records after update
SELECT *
FROM Patients;
-- Rollback to undo the update
ROLLBACK;
SELECT * FROM patients;
```



➢

Output after Rollback Transaction

```
108  -- Rollback to undo the update
109  ROLLBACK;
110  SELECT * FROM patients;
111
112  --Query 13: Advanced Query 1---This query combines joins and a subquery to calculate total appointme
113
114  SELECT d.Name AS DoctorName,
115         subquery.TotalAppointments,
116         subquery.TotalRevenue
117  FROM Doctor d
```

Data Output | Messages | Notifications

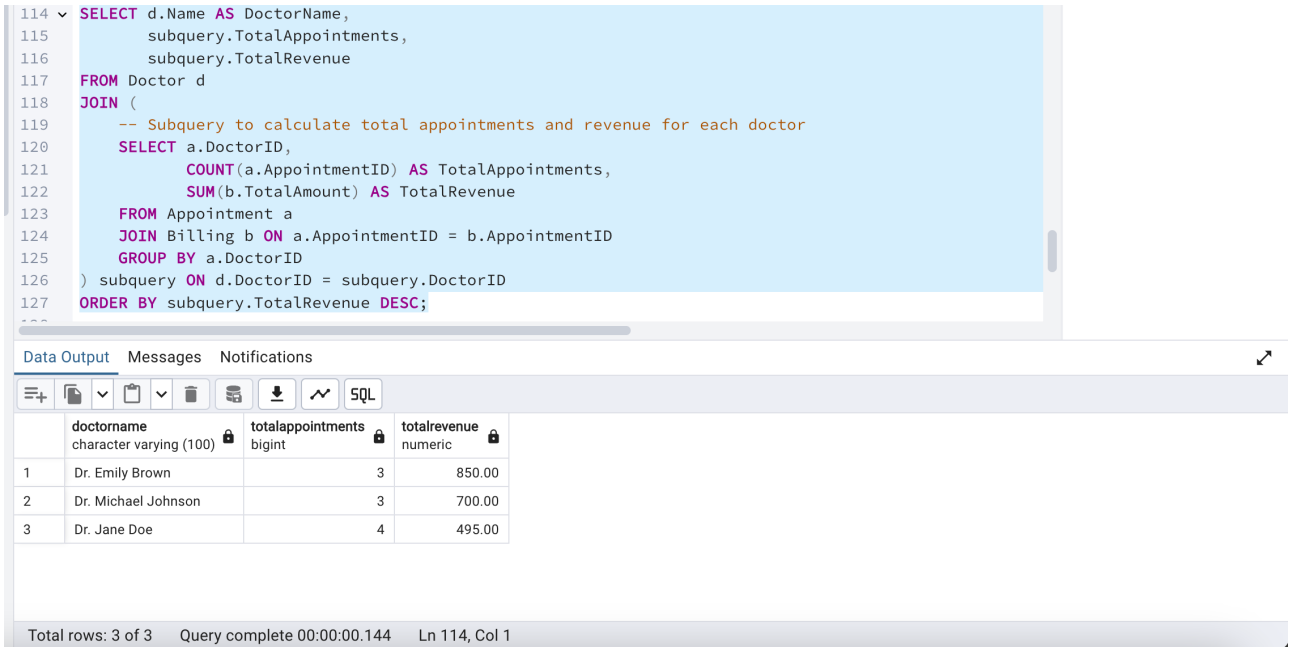
	patientid [PK] integer	name character varying (100)	dateofbirth date	gender character (1)	contactinfo character varying (255)	address character varying (255)	medicalhistory text
1	1	John Doe	1985-01-15	M	johndoe@example.com	123 Main St	Allergic to peanuts
2	2	Jane Smith	1990-07-22	F	janesmith@example.com	456 Oak St	No known allergies
3	3	Michael Johnson	1978-11-09	M	michaeljohnson@example.com	789 Elm St	High blood pressure
4	4	Emily Brown	1992-03-12	F	emilybrown@example.com	101 Pine St	Asthma
5	5	David Lee	1988-09-25	M	davidlee@example.com	222 Cedar St	Diabetes
6	6	Sarah Kim	1995-05-18	F	sarahkim@example.com	333 Maple St	Migraines
7	7	Christopher Wilson	1981-11-07	M	christopherwilson@example.com	444 Oak St	Heart condition
8	8	Olivia Taylor	1993-02-20	F	oliviataylor@example.com	555 Pine St	Food allergies

Total rows: 30 of 30 Query complete 00:00:00.122 Ln 110, Col 1

➢

Query 13: Show the total number of appointments and total revenue generated by each doctor order by revenue

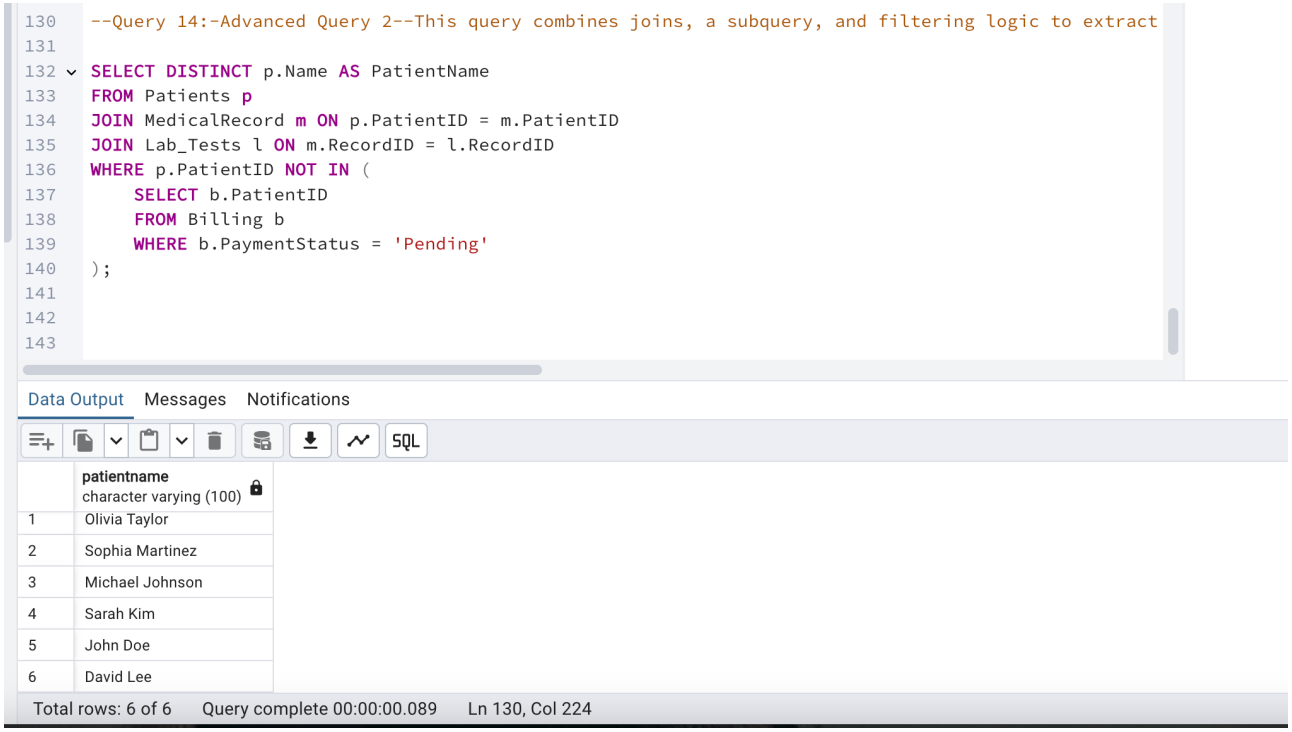
```
SELECT d.Name AS DoctorName,
      subquery.TotalAppointments,
      subquery.TotalRevenue
FROM Doctor d
JOIN (
   -- Subquery to calculate total appointments and revenue for each doctor
   SELECT a.DoctorID,
         COUNT(a.AppointmentID) AS TotalAppointments,
         SUM(b.TotalAmount) AS TotalRevenue
   FROM Appointment a
   JOIN Billing b ON a.AppointmentID = b.AppointmentID
   GROUP BY a.DoctorID
) subquery ON d.DoctorID = subquery.DoctorID
ORDER BY subquery.TotalRevenue DESC;
```



➢

Query 14: identify the patients who have completed all their payments and are not associated with any pending payment status in the billing system

```
SELECT DISTINCT p.Name AS PatientName
FROM Patients p
JOIN MedicalRecord m ON p.PatientID = m.PatientID
JOIN Lab_Tests l ON m.RecordID = l.RecordID
WHERE p.PatientID NOT IN (
   SELECT b.PatientID
   FROM Billing b
   WHERE b.PaymentStatus = 'Pending'
);
```



```
130  --Query 14:-Advanced Query 2--This query combines joins, a subquery, and filtering logic to extract
131
132  SELECT DISTINCT p.Name AS PatientName
133  FROM Patients p
134  JOIN MedicalRecord m ON p.PatientID = m.PatientID
135  JOIN Lab_Tests l ON m.RecordID = l.RecordID
136  WHERE p.PatientID NOT IN (
137      SELECT b.PatientID
138      FROM Billing b
139      WHERE b.PaymentStatus = 'Pending'
140  );
141
142
143
```

Data Output | Messages | Notifications

	patientname character varying (100)
1	Olivia Taylor
2	Sophia Martinez
3	Michael Johnson
4	Sarah Kim
5	John Doe
6	David Lee

Total rows: 6 of 6 Query complete 00:00:00.089 Ln 130, Col 224

4. Conclusion:

In conclusion, the database design of the Health Care Management System is crucial for smooth and efficient operations. It helps manage patient records, appointments, and billing effectively while ensuring data security. A good database design supports better healthcare delivery and provides a strong foundation for future improvements.