SCHEMA CREATION AND SQL QUERIES

DBMS PHASE 4

GROUP - 1

VISHNU TEJAS E [AM.SC.U4AIE23164]
SOURI S [AM.SC.U4AIE23151]
NISHANT SHAH [AM.SC.U4AIE23166]
ARJUN K INDUNESH [AM.SC.U4AIE23109]

1. Create Tables with necessary constraints pertaining to the relational schema using appropriate DDL statements

```
-- Function to automatically update updated_at timestamp
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = CURRENT_TIMESTAMP;
    RETURN NEW;
END;
$$ language 'plpgsql';


CREATE TABLE "roles" (
  "role_id" int PRIMARY KEY,
  "role_name" varchar
);


CREATE TABLE "permissions" (
  "permission_id" int PRIMARY KEY,
  "permission_name" varchar,
  "description" text
);


CREATE TABLE "role_permissions" (
  "role_id" int,
  "permission_id" int,
  PRIMARY KEY ("role_id", "permission_id")
);
```

```
CREATE TABLE "users" (
  "user_id" int PRIMARY KEY,
  "email" varchar,
  "password_hash" varchar,
  "is_active" boolean,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP,
  "updated_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "user_roles" (
  "user_id" int,
  "role_id" int,
  PRIMARY KEY ("user_id", "role_id")
);

CREATE TABLE "staff_profiles" (
  "staff_id" int PRIMARY KEY,
  "user_id" int,
  "first_name" varchar,
  "last_name" varchar,
  "contact_email" varchar,
  "emergency_contact" varchar,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP,
  "updated_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "clinics" (
  "clinic_id" int PRIMARY KEY,
  "clinic_name" varchar,
  "clinic_address" varchar,
  "city" varchar,
  "state" varchar,
  "postal_code" varchar,
  "weekday_hours" varchar,
  "weekend_hours" varchar
);
```

```
CREATE TABLE "patients" (
  "patient_id" int PRIMARY KEY,
  "primary_clinic_id" int,
  "first_name" varchar,
  "last_name" varchar,
  "date_of_birth" date,
  "gender" varchar,
  "street_address" varchar,
  "city" varchar,
  "state" varchar,
  "postal_code" varchar,
  "country" varchar,
  "registration_date" date,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP,
  "updated_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "emergency_contacts" (
  "emergency_contact_id" int PRIMARY KEY,
  "patient_id" int,
  "full_name" varchar,
  "contact_relationship" varchar,
  "phone_number" varchar,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "medical_history" (
  "medical_history_id" int PRIMARY KEY,
  "patient_id" int,
  "history_type" varchar,
  "description" text,
  "start_date" date,
  "end_date" date,
  "is_active" boolean,
  "recorded_at" timestamp
```

```
);

CREATE TABLE "provider_schedules" (
  "schedule_id" int PRIMARY KEY,
  "staff_id" int,
  "clinic_id" int,
  "schedule_rules" text,
  "valid_from" date,
  "valid_to" date
);

CREATE TABLE "appointments" (
  "appointment_id" int PRIMARY KEY,
  "patient_id" int,
  "staff_id" int,
  "clinic_id" int,
  "appointment_period" tsrange,
  "appointment_type" varchar,
  "status" varchar,
  "notes" text,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP,
  "updated_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "treatment_codes" (
  "code_id" int PRIMARY KEY,
  "code_system" varchar,
  "code_value" varchar,
  "description" text,
  "standard_fee" decimal,
  "is_active" boolean
);

CREATE TABLE "treatment_plans" (
  "plan_id" int PRIMARY KEY,
  "patient_id" int,
```

```
  "staff_id" int,
  "plan_date" date,
  "description" text,
  "status" varchar,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "plan_items" (
  "plan_item_id" int PRIMARY KEY,
  "plan_id" int,
  "code_id" int,
  "quantity" int,
  "fee_quoted" decimal
);

CREATE TABLE "procedures_log" (
  "procedure_id" int PRIMARY KEY,
  "appointment_id" int,
  "patient_id" int,
  "staff_id" int,
  "code_id" int,
  "procedure_date" date,
  "fee_charged" decimal,
  "notes" text
);

CREATE TABLE "invoices" (
  "invoice_id" int PRIMARY KEY,
  "patient_id" int,
  "invoice_date" date,
  "due_date" date,
  "total_amount" decimal,
  "amount_paid" decimal,
  "status" varchar,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);
```

```
CREATE TABLE "invoice_items" (
  "invoice_item_id" int PRIMARY KEY,
  "invoice_id" int,
  "procedure_id" int,
  "amount" decimal
);

CREATE TABLE "payments" (
  "payment_id" int PRIMARY KEY,
  "invoice_id" int,
  "payment_date" date,
  "amount" decimal,
  "payment_method" varchar,
  "transaction_id" varchar,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "insurance_companies" (
  "company_id" int PRIMARY KEY,
  "company_name" varchar,
  "payer_id" varchar,
  "contact_info" text
);

CREATE TABLE "patient_plans" (
  "patient_plan_id" int PRIMARY KEY,
  "patient_id" int,
  "company_id" int,
  "policy_number" varchar,
  "group_number" varchar,
  "subscriber_id" varchar,
  "relationship_to_subscriber" varchar,
  "is_primary" boolean,
  "effective_date" date,
  "termination_date" date
```

```
);

CREATE TABLE "claims" (
  "claim_id" int PRIMARY KEY,
  "invoice_id" int,
  "patient_plan_id" int,
  "submission_date" date,
  "status" varchar,
  "paid_amount" decimal,
  "adjudication_date" date
);

CREATE TABLE "teeth" (
  "tooth_id" int PRIMARY KEY,
  "universal_number" varchar,
  "tooth_name" varchar,
  "is_primary" boolean
);

CREATE TABLE "patient_odontogram" (
  "patient_id" int,
  "tooth_id" int,
  "current_state" varchar,
  "state_history" text,
  "updated_at" timestamp DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY ("patient_id", "tooth_id")
);

CREATE TABLE "perio_exams" (
  "perio_exam_id" int PRIMARY KEY,
  "patient_id" int,
  "staff_id" int,
  "exam_date" date,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);
```

```
CREATE TABLE "perio_measurements" (
  "perio_measurement_id" int PRIMARY KEY,
  "perio_exam_id" int,
  "tooth_id" int,
  "measurement_type" varchar,
  "site" varchar,
  "value" varchar,
  "exam_date" date
);

CREATE TABLE "prescriptions" (
  "prescription_id" int PRIMARY KEY,
  "patient_id" int,
  "staff_id" int,
  "appointment_id" int,
  "drug_name" varchar,
  "dosage" varchar,
  "quantity" int,
  "instructions" text,
  "refills" int,
  "date_prescribed" date,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "clinical_notes" (
  "note_id" int PRIMARY KEY,
  "appointment_id" int,
  "patient_id" int,
  "staff_id" int,
  "note_title" varchar,
  "note_text" text,
  "is_finalized" boolean,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP,
  "updated_at" timestamp DEFAULT CURRENT_TIMESTAMP
);
```

```
CREATE TABLE "documents" (
  "document_id" int PRIMARY KEY,
  "patient_id" int,
  "appointment_id" int,
  "document_type" varchar,
  "file_name" varchar,
  "storage_identifier" varchar,
  "mime_type" varchar,
  "file_size_bytes" int,
  "metadata" text,
  "upload_timestamp" timestamp
);

CREATE TABLE "dentists" (
  "staff_id" int PRIMARY KEY,
  "dental_degree" varchar,
  "years_of_experience" int
);

CREATE TABLE "administrative_staff" (
  "staff_id" int PRIMARY KEY,
  "department" varchar,
  "access_level" varchar
);

CREATE TABLE "nurses" (
  "staff_id" int PRIMARY KEY,
  "nursing_license_number" varchar,
  "nursing_degree" varchar
);

CREATE TABLE "healthcare_providers" (
  "provider_id" int PRIMARY KEY,
  "provider_type" varchar,
  "provider_ref_id" int
);
```

```
CREATE TABLE "external_specialists" (
  "specialist_id" int PRIMARY KEY,
  "specialty_area" varchar,
  "referral_network" text
);

CREATE TABLE "staff_specializations" (
  "specialization_id" int PRIMARY KEY,
  "staff_id" int,
  "specialization" varchar,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "staff_phone_numbers" (
  "phone_id" int PRIMARY KEY,
  "staff_id" int,
  "phone_number" varchar,
  "phone_type" varchar,
  "is_primary" boolean,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "clinic_phone_numbers" (
  "phone_id" int PRIMARY KEY,
  "clinic_id" int,
  "phone_number" varchar,
  "phone_type" varchar,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "clinic_email_addresses" (
  "email_id" int PRIMARY KEY,
  "clinic_id" int,
  "email_address" varchar,
  "email_type" varchar,
```

```
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "clinic_services" (
  "service_id" int PRIMARY KEY,
  "clinic_id" int,
  "service_name" varchar,
  "description" text,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "clinic_insurance_accepted" (
  "insurance_id" int PRIMARY KEY,
  "clinic_id" int,
  "insurance_name" varchar,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "patient_allergies" (
  "allergy_id" int PRIMARY KEY,
  "patient_id" int,
  "allergy_name" varchar,
  "severity" varchar,
  "notes" text,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "patient_email_addresses" (
  "email_id" int PRIMARY KEY,
  "patient_id" int,
  "email_address" varchar,
  "email_type" varchar,
  "is_primary" boolean,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);
```

```
CREATE TABLE "patient_phone_numbers" (
  "phone_id" int PRIMARY KEY,
  "patient_id" int,
  "phone_number" varchar,
  "phone_type" varchar,
  "is_primary" boolean,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "dentist_procedures" (
  "procedure_id" int PRIMARY KEY,
  "staff_id" int,
  "procedure_name" varchar,
  "proficiency_level" varchar,
  "years_experience" int,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "admin_responsibilities" (
  "responsibility_id" int PRIMARY KEY,
  "staff_id" int,
  "responsibility" varchar,
  "description" text,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE "nurse_specializations" (
  "specialization_id" int PRIMARY KEY,
  "staff_id" int,
  "specialization" varchar,
  "certification_date" date,
  "certification_expiry" date,
  "created_at" timestamp DEFAULT CURRENT_TIMESTAMP
);

ALTER TABLE "staff_profiles" ADD FOREIGN KEY ("user_id") REFERENCES "users" ("user_id");
```

```
ALTER TABLE "user_roles" ADD FOREIGN KEY ("user_id") REFERENCES "users" ("user_id");

ALTER TABLE "user_roles" ADD FOREIGN KEY ("role_id") REFERENCES "roles" ("role_id");

ALTER TABLE "role_permissions" ADD FOREIGN KEY ("role_id") REFERENCES "roles" ("role_id");

ALTER TABLE "role_permissions" ADD FOREIGN KEY ("permission_id") REFERENCES "permissions"
("permission_id");

ALTER TABLE "patients" ADD FOREIGN KEY ("primary_clinic_id") REFERENCES "clinics" ("clinic_id");

ALTER TABLE "emergency_contacts" ADD FOREIGN KEY ("patient_id") REFERENCES "patients"
("patient_id");

ALTER TABLE "medical_history" ADD FOREIGN KEY ("patient_id") REFERENCES "patients"
("patient_id");

ALTER TABLE "provider_schedules" ADD FOREIGN KEY ("staff_id") REFERENCES "staff_profiles"
("staff_id");

ALTER TABLE "provider_schedules" ADD FOREIGN KEY ("clinic_id") REFERENCES "clinics"
("clinic_id");

ALTER TABLE "appointments" ADD FOREIGN KEY ("patient_id") REFERENCES "patients" ("patient_id");

ALTER TABLE "appointments" ADD FOREIGN KEY ("staff_id") REFERENCES "staff_profiles" ("staff_id");

ALTER TABLE "appointments" ADD FOREIGN KEY ("clinic_id") REFERENCES "clinics" ("clinic_id");

ALTER TABLE "plan_items" ADD FOREIGN KEY ("plan_id") REFERENCES "treatment_plans" ("plan_id");

ALTER TABLE "plan_items" ADD FOREIGN KEY ("code_id") REFERENCES "treatment_codes"
("code_id");

ALTER TABLE "treatment_plans" ADD FOREIGN KEY ("patient_id") REFERENCES "patients"
("patient_id");
```

```
ALTER TABLE "treatment_plans" ADD FOREIGN KEY ("staff_id") REFERENCES "staff_profiles" ("staff_id");

ALTER TABLE "procedures_log" ADD FOREIGN KEY ("appointment_id") REFERENCES "appointments" ("appointment_id");

ALTER TABLE "procedures_log" ADD FOREIGN KEY ("patient_id") REFERENCES "patients" ("patient_id");

ALTER TABLE "procedures_log" ADD FOREIGN KEY ("staff_id") REFERENCES "staff_profiles" ("staff_id");

ALTER TABLE "procedures_log" ADD FOREIGN KEY ("code_id") REFERENCES "treatment_codes" ("code_id");

ALTER TABLE "invoice_items" ADD FOREIGN KEY ("invoice_id") REFERENCES "invoices" ("invoice_id");

ALTER TABLE "invoice_items" ADD FOREIGN KEY ("procedure_id") REFERENCES "procedures_log" ("procedure_id");

ALTER TABLE "invoices" ADD FOREIGN KEY ("patient_id") REFERENCES "patients" ("patient_id");

ALTER TABLE "payments" ADD FOREIGN KEY ("invoice_id") REFERENCES "invoices" ("invoice_id");

ALTER TABLE "patient_plans" ADD FOREIGN KEY ("patient_id") REFERENCES "patients" ("patient_id");

ALTER TABLE "patient_plans" ADD FOREIGN KEY ("company_id") REFERENCES "insurance_companies" ("company_id");

ALTER TABLE "claims" ADD FOREIGN KEY ("invoice_id") REFERENCES "invoices" ("invoice_id");

ALTER TABLE "claims" ADD FOREIGN KEY ("patient_plan_id") REFERENCES "patient_plans" ("patient_plan_id");

ALTER TABLE "patient_odontogram" ADD FOREIGN KEY ("patient_id") REFERENCES "patients" ("patient_id");

ALTER TABLE "patient_odontogram" ADD FOREIGN KEY ("tooth_id") REFERENCES "teeth" ("tooth_id");
```

```
ALTER TABLE "perio_exams" ADD FOREIGN KEY ("patient_id") REFERENCES "patients" ("patient_id");

ALTER TABLE "perio_exams" ADD FOREIGN KEY ("staff_id") REFERENCES "staff_profiles" ("staff_id");

ALTER TABLE "perio_measurements" ADD FOREIGN KEY ("perio_exam_id") REFERENCES
"perio_exams" ("perio_exam_id");

ALTER TABLE "perio_measurements" ADD FOREIGN KEY ("tooth_id") REFERENCES "teeth" ("tooth_id");

ALTER TABLE "prescriptions" ADD FOREIGN KEY ("patient_id") REFERENCES "patients" ("patient_id");

ALTER TABLE "prescriptions" ADD FOREIGN KEY ("staff_id") REFERENCES "staff_profiles" ("staff_id");

ALTER TABLE "prescriptions" ADD FOREIGN KEY ("appointment_id") REFERENCES "appointments"
("appointment_id");

ALTER TABLE "clinical_notes" ADD FOREIGN KEY ("appointment_id") REFERENCES "appointments"
("appointment_id");

ALTER TABLE "clinical_notes" ADD FOREIGN KEY ("patient_id") REFERENCES "patients" ("patient_id");

ALTER TABLE "clinical_notes" ADD FOREIGN KEY ("staff_id") REFERENCES "staff_profiles" ("staff_id");

ALTER TABLE "documents" ADD FOREIGN KEY ("patient_id") REFERENCES "patients" ("patient_id");

ALTER TABLE "documents" ADD FOREIGN KEY ("appointment_id") REFERENCES "appointments"
("appointment_id");

ALTER TABLE "staff_specializations" ADD FOREIGN KEY ("staff_id") REFERENCES "staff_profiles"
("staff_id");

ALTER TABLE "staff_phone_numbers" ADD FOREIGN KEY ("staff_id") REFERENCES "staff_profiles"
("staff_id");

ALTER TABLE "clinic_phone_numbers" ADD FOREIGN KEY ("clinic_id") REFERENCES "clinics"
("clinic_id");

ALTER TABLE "clinic_email_addresses" ADD FOREIGN KEY ("clinic_id") REFERENCES "clinics"
("clinic_id");
```

```
ALTER TABLE "clinic_services" ADD FOREIGN KEY ("clinic_id") REFERENCES "clinics" ("clinic_id");

ALTER TABLE "clinic_insurance_accepted" ADD FOREIGN KEY ("clinic_id") REFERENCES "clinics"
("clinic_id");

ALTER TABLE "patient_allergies" ADD FOREIGN KEY ("patient_id") REFERENCES "patients"
("patient_id");

ALTER TABLE "patient_email_addresses" ADD FOREIGN KEY ("patient_id") REFERENCES "patients"
("patient_id");

ALTER TABLE "patient_phone_numbers" ADD FOREIGN KEY ("patient_id") REFERENCES "patients"
("patient_id");

ALTER TABLE "dentist_procedures" ADD FOREIGN KEY ("staff_id") REFERENCES "dentists" ("staff_id");

ALTER TABLE "admin_responsibilities" ADD FOREIGN KEY ("staff_id") REFERENCES
"administrative_staff" ("staff_id");

ALTER TABLE "nurse_specializations" ADD FOREIGN KEY ("staff_id") REFERENCES "nurses"
("staff_id");

ALTER TABLE "dentists" ADD FOREIGN KEY ("staff_id") REFERENCES "staff_profiles" ("staff_id");

ALTER TABLE "administrative_staff" ADD FOREIGN KEY ("staff_id") REFERENCES "staff_profiles"
("staff_id");

ALTER TABLE "nurses" ADD FOREIGN KEY ("staff_id") REFERENCES "staff_profiles" ("staff_id");

ALTER TABLE "healthcare_providers" ADD FOREIGN KEY ("provider_ref_id") REFERENCES
"staff_profiles" ("staff_id");

ALTER TABLE "healthcare_providers" ADD FOREIGN KEY ("provider_ref_id") REFERENCES
"external_specialists" ("specialist_id");

-- Create triggers to automatically update updated_at timestamps

CREATE TRIGGER update_users_updated_at BEFORE UPDATE ON "users" FOR EACH ROW EXECUTE
FUNCTION update_updated_at_column();
```

CREATE TRIGGER update_staff_profiles_updated_at BEFORE UPDATE ON "staff_profiles" FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_patients_updated_at BEFORE UPDATE ON "patients" FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_appointments_updated_at BEFORE UPDATE ON "appointments" FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_patient_odontogram_updated_at BEFORE UPDATE ON "patient_odontogram" FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_clinical_notes_updated_at BEFORE UPDATE ON "clinical_notes" FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

2. Populate tables with few tuples.

-- Round 1: Tables with no (or self-referencing) foreign keys

INSERT INTO "roles" ("role_id", "role_name") VALUES

(1, 'Admin'),

(2, 'Dentist'),

(3, 'Nurse'),

(4, 'Patient'),

(5, 'Receptionist');

INSERT INTO "permissions" ("permission_id", "permission_name", "description") VALUES

(1, 'manage_users', 'Can create, edit, and delete users.'),

(2, 'manage_patients', 'Can create, edit, and view patient records.'),

(3, 'manage_appointments', 'Can schedule and modify appointments.'),

(4, 'view_billing', 'Can view invoices and payments.'),

(5, 'manage_billing', 'Can create and modify invoices and payments.'),

(6, 'view_clinical_notes', 'Can view clinical notes.'),

(7, 'manage_clinical_notes', 'Can create and edit clinical notes.');

INSERT INTO "users" ("user_id", "email", "password_hash", "is_active", "created_at", "updated_at") VALUES

(1, 'admin@dental.com', 'hashed_password_placeholder_123', true, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),

(2, 'dr.eve@dental.com', 'hashed_password_placeholder_456', true, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),

(3, 'nurse.bob@dental.com', 'hashed_password_placeholder_789', true, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),

(4, 'reception@dental.com', 'hashed_password_placeholder_101', true, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),

(5, 'patient.alice@example.com', 'hashed_password_placeholder_112', true, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),

(6, 'patient.charlie@example.com', 'hashed_password_placeholder_113', true, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP);

-- Note: User IDs for patients are not strictly necessary if patients don't log in.

INSERT INTO "clinics" ("clinic_id", "clinic_name", "clinic_address", "city", "state", "postal_code", "weekday_hours", "weekend_hours") VALUES

(1, 'Downtown Dental Care', '123 Main St', 'Metropolis', 'NY', '10001', '8:00 AM - 6:00 PM', '9:00 AM - 1:00 PM'),

(2, 'Uptown Smiles', '456 Oak Ave', 'Metropolis', 'NY', '10021', '9:00 AM - 5:00 PM', 'Closed');

INSERT INTO "treatment_codes" ("code_id", "code_system", "code_value", "description", "standard_fee", "is_active") VALUES

(1, 'ADA', 'D0120', 'Periodic oral evaluation', 55.00, true),

(2, 'ADA', 'D1110', 'Prophylaxis - adult', 120.00, true),

(3, 'ADA', 'D2391', 'Resin-based composite - one surface, posterior', 250.00, true),

(4, 'ADA', 'D0220', 'Intraoral - periapical first film', 30.00, true);

INSERT INTO "insurance_companies" ("company_id", "company_name", "payer_id", "contact_info") VALUES

(1, 'MetroHealth Insurance', 'MH12345', '1-800-555-1234, claims@metrohealth.com'),

(2, 'DentalGuard', 'DG98765', '1-800-555-6789, providers@dentalguard.com');

INSERT INTO "teeth" ("tooth_id", "universal_number", "tooth_name", "is_primary") VALUES

(1, '1', 'Third Molar - Upper Right', false),

(2, '2', 'Second Molar - Upper Right', false),

(3, '3', 'First Molar - Upper Right', false),

(4, '4', 'Second Bicuspid - Upper Right', false),

(5, '5', 'First Bicuspid - Upper Right', false),

(6, '6', 'Cuspid (Canine) - Upper Right', false),

(7, '7', 'Lateral Incisor - Upper Right', false),

(8, '8', 'Central Incisor - Upper Right', false),

(9, '9', 'Central Incisor - Upper Left', false),

(10, '10', 'Lateral Incisor - Upper Left', false),

(11, '11', 'Cuspid (Canine) - Upper Left', false),

(12, '12', 'First Bicuspid - Upper Left', false),

(13, '13', 'Second Bicuspid - Upper Left', false),

(14, '14', 'First Molar - Upper Left', false),

(15, '15', 'Second Molar - Upper Left', false),

(16, '16', 'Third Molar - Upper Left', false),
(17, '17', 'Third Molar - Lower Left', false),
(18, '18', 'Second Molar - Lower Left', false),
(19, '19', 'First Molar - Lower Left', false),
(20, '20', 'Second Bicuspid - Lower Left', false),
(21, '21', 'First Bicuspid - Lower Left', false),
(22, '22', 'Cuspid (Canine) - Lower Left', false),
(23, '23', 'Lateral Incisor - Lower Left', false),
(24, '24', 'Central Incisor - Lower Left', false),
(25, '25', 'Central Incisor - Lower Right', false),
(26, '26', 'Lateral Incisor - Lower Right', false),
(27, '27', 'Cuspid (Canine) - Lower Right', false),
(28, '28', 'First Bicuspid - Lower Right', false),
(29, '29', 'Second Bicuspid - Lower Right', false),
(30, '30', 'First Molar - Lower Right', false),
(31, '31', 'Second Molar - Lower Right', false),
(32, '32', 'Third Molar - Lower Right', false);

INSERT INTO "external_specialists" ("specialist_id", "specialty_area", "referral_network") VALUES
(1, 'Orthodontics', 'Metropolis Specialist Network'),
(2, 'Endodontics', 'City Endodontics Group');

-- Round 2: Tables dependent on Round 1

INSERT INTO "role_permissions" ("role_id", "permission_id") VALUES
(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), -- Admin: all
(2, 2), (2, 3), (2, 6), (2, 7), -- Dentist: manage patients, appts, clinical notes
(3, 2), (3, 6), -- Nurse: manage patients, view clinical notes
(5, 2), (5, 3), (5, 4); -- Receptionist: manage patients, appts, view billing

INSERT INTO "user_roles" ("user_id", "role_id") VALUES
(1, 1), -- admin@dental.com is Admin
(2, 2), -- dr.eve@dental.com is Dentist
(3, 3), -- nurse.bob@dental.com is Nurse
(4, 5); -- reception@dental.com is Receptionist
-- Note: Patients (user_id 5, 6) could be given role_id 4

INSERT INTO "staff_profiles" ("staff_id", "user_id", "first_name", "last_name", "contact_email", "emergency_contact", "created_at", "updated_at") VALUES

(1, 2, 'Eve', 'Hanson', 'dr.eve@dental.com', 'Mark Hanson (Spouse) 555-0101', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),

(2, 3, 'Bob', 'Miller', 'nurse.bob@dental.com', 'Susan Miller (Sister) 555-0202', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),

(3, 4, 'Sarah', 'Chen', 'reception@dental.com', 'Tom Chen (Father) 555-0303', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),

(4, 1, 'Adam', 'Admin', 'admin@dental.com', 'N/A', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP);

INSERT INTO "patients" ("patient_id", "primary_clinic_id", "first_name", "last_name", "date_of_birth", "gender", "street_address", "city", "state", "postal_code", "country", "registration_date", "created_at", "updated_at") VALUES

(1, 1, 'Alice', 'Smith', '1990-05-15', 'Female', '789 Pine Ln', 'Metropolis', 'NY', '10005', 'USA', '2023-01-10', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),

(2, 1, 'Charlie', 'Brown', '1985-11-22', 'Male', '321 Elm St', 'Metropolis', 'NY', '10003', 'USA', '2022-11-05', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),

(3, 2, 'David', 'Lee', '2005-02-28', 'Male', '654 Maple Dr', 'Metropolis', 'NY', '10021', 'USA', '2023-03-20', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP);

-- Round 3: Tables dependent on Round 2

INSERT INTO "staff_phone_numbers" ("phone_id", "staff_id", "phone_number", "phone_type", "is_primary", "created_at") VALUES

(1, 1, '555-111-2222', 'Work', true, CURRENT_TIMESTAMP),

(2, 2, '555-111-3333', 'Work', true, CURRENT_TIMESTAMP),

(3, 3, '555-111-4444', 'Work', true, CURRENT_TIMESTAMP);

INSERT INTO "clinic_phone_numbers" ("phone_id", "clinic_id", "phone_number", "phone_type", "created_at") VALUES

(1, 1, '212-555-1000', 'Main', CURRENT_TIMESTAMP),

(2, 1, '212-555-1001', 'Fax', CURRENT_TIMESTAMP),

(3, 2, '212-555-2000', 'Main', CURRENT_TIMESTAMP);

INSERT INTO "clinic_email_addresses" ("email_id", "clinic_id", "email_address", "email_type", "created_at") VALUES

(1, 1, 'info@downtowndental.com', 'General', CURRENT_TIMESTAMP),

(2, 1, 'billing@downtowndental.com', 'Billing', CURRENT_TIMESTAMP),

(3, 2, 'contact@uptownsmiles.com', 'General', CURRENT_TIMESTAMP);

INSERT INTO "clinic_services" ("service_id", "clinic_id", "service_name", "description", "created_at") VALUES

(1, 1, 'General Dentistry', 'Checkups, cleanings, fillings.', CURRENT_TIMESTAMP),

(2, 1, 'Cosmetic Dentistry', 'Whitening, veneers.', CURRENT_TIMESTAMP),

(3, 2, 'General Dentistry', 'Checkups, cleanings, fillings.', CURRENT_TIMESTAMP);

INSERT INTO "clinic_insurance_accepted" ("insurance_id", "clinic_id", "insurance_name", "created_at") VALUES

(1, 1, 'MetroHealth Insurance', CURRENT_TIMESTAMP),

(2, 1, 'DentalGuard', CURRENT_TIMESTAMP),

(3, 2, 'MetroHealth Insurance', CURRENT_TIMESTAMP);

INSERT INTO "emergency_contacts" ("emergency_contact_id", "patient_id", "full_name", "contact_relationship", "phone_number", "created_at") VALUES

(1, 1, 'John Smith', 'Spouse', '555-0123', CURRENT_TIMESTAMP),

(2, 2, 'Sally Brown', 'Sister', '555-0456', CURRENT_TIMESTAMP),

(3, 3, 'Grace Lee', 'Mother', '555-0789', CURRENT_TIMESTAMP);

INSERT INTO "medical_history" ("medical_history_id", "patient_id", "history_type", "description", "start_date", "end_date", "is_active", "recorded_at") VALUES

(1, 1, 'Allergy', 'Penicillin', '1995-01-01', NULL, true, '2023-01-10 09:00:00'),

(2, 1, 'Condition', 'Hypertension', '2018-06-15', NULL, true, '2023-01-10 09:00:00'),

(3, 2, 'Surgery', 'Appendectomy', '2010-07-20', '2010-07-20', false, '2022-11-05 14:00:00');

INSERT INTO "patient_allergies" ("allergy_id", "patient_id", "allergy_name", "severity", "notes", "created_at") VALUES

(1, 1, 'Penicillin', 'Severe', 'Anaphylaxis', CURRENT_TIMESTAMP),

(2, 1, 'Latex', 'Mild', 'Skin rash', CURRENT_TIMESTAMP),

(3, 3, 'Peanuts', 'Severe', 'Carries EpiPen', CURRENT_TIMESTAMP);

INSERT INTO "patient_email_addresses" ("email_id", "patient_id", "email_address", "email_type", "is_primary", "created_at") VALUES

(1, 1, 'alice.smith@example.com', 'Personal', true, CURRENT_TIMESTAMP),

(2, 2, 'cbrown@example.com', 'Personal', true, CURRENT_TIMESTAMP),

(3, 3, 'david.lee@example.com', 'Personal', true, CURRENT_TIMESTAMP);

INSERT INTO "patient_phone_numbers" ("phone_id", "patient_id", "phone_number", "phone_type", "is_primary", "created_at") VALUES

(1, 1, '555-321-1234', 'Mobile', true, CURRENT_TIMESTAMP),

(2, 2, '555-654-4567', 'Mobile', true, CURRENT_TIMESTAMP),

(3, 2, '555-654-9999', 'Work', false, CURRENT_TIMESTAMP),

(4, 3, '555-987-7890', 'Mobile', true, CURRENT_TIMESTAMP);

INSERT INTO "provider_schedules" ("schedule_id", "staff_id", "clinic_id", "schedule_rules", "valid_from", "valid_to") VALUES

(1, 1, 1, '{"Mon": "8:00-17:00", "Tue": "8:00-17:00", "Wed": "9:00-13:00"}', '2023-01-01', '2023-12-31'),

(2, 2, 1, '{"Mon": "8:00-17:00", "Tue": "8:00-17:00"}', '2023-01-01', '2023-12-31'),

(3, 1, 2, '{"Thu": "9:00-17:00", "Fri": "9:00-17:00"}', '2023-01-01', '2023-12-31');

INSERT INTO "treatment_plans" ("plan_id", "patient_id", "staff_id", "plan_date", "description", "status", "created_at") VALUES

(1, 1, 1, '2023-01-15', 'Initial treatment plan for Alice', 'Active', CURRENT_TIMESTAMP),

(2, 2, 1, '2022-11-10', 'Restorative work for Charlie', 'Completed', CURRENT_TIMESTAMP);

INSERT INTO "dentists" ("staff_id", "dental_degree", "years_of_experience") VALUES

(1, 'DDS', 10);

INSERT INTO "administrative_staff" ("staff_id", "department", "access_level") VALUES

(3, 'Front Desk', 'Level 2'),

(4, 'Management', 'Level 5');

INSERT INTO "nurses" ("staff_id", "nursing_license_number", "nursing_degree") VALUES

(2, 'RN123456', 'RN');

INSERT INTO "patient_plans" ("patient_plan_id", "patient_id", "company_id", "policy_number", "group_number", "subscriber_id", "relationship_to_subscriber", "is_primary", "effective_date", "termination_date") VALUES

(1, 1, 1, 'P123456789', 'G654321', 'S-12345', 'Self', true, '2023-01-01', NULL),

(2, 2, 2, 'P987654321', 'G123456', 'S-67890', 'Self', true, '2022-01-01', NULL),

(3, 3, 1, 'P123456789', 'G654321', 'S-98765', 'Dependent', true, '2020-01-01', NULL); -- David Lee, dependent on parent's plan

INSERT INTO "patient_odontogram" ("patient_id", "tooth_id", "current_state", "state_history", "updated_at") VALUES

(1, 3, 'Restored (Composite)', '[{"date": "2023-02-01", "state": "Restored (Composite)"}]', CURRENT_TIMESTAMP),

(1, 16, 'Missing', '[{"date": "2015-01-01", "state": "Missing"}]', CURRENT_TIMESTAMP),

(2, 30, 'Caries', '[{"date": "2022-11-10", "state": "Caries"}]', CURRENT_TIMESTAMP);

INSERT INTO "perio_exams" ("perio_exam_id", "patient_id", "staff_id", "exam_date", "created_at") VALUES

(1, 1, 1, '2023-01-15', CURRENT_TIMESTAMP),

(2, 2, 1, '2022-11-10', CURRENT_TIMESTAMP);

-- Round 4: Tables dependent on Round 3

INSERT INTO "appointments" ("appointment_id", "patient_id", "staff_id", "clinic_id", "appointment_period", "appointment_type", "status", "notes", "created_at", "updated_at") VALUES

(1, 1, 1, 1, '[2023-01-15 09:00:00, 2023-01-15 10:00:00)', 'New Patient Exam', 'Completed', 'New patient, full exam and x-rays.', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),

(2, 2, 1, 1, '[2022-11-10 14:00:00, 2022-11-10 14:30:00)', 'Periodic Exam', 'Completed', '6-month checkup.', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),

(3, 1, 1, 1, '[2023-02-01 11:00:00, 2023-02-01 12:00:00)', 'Restorative', 'Completed', 'Filling on #3.', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),

(4, 3, 1, 2, '[2023-03-20 10:00:00, 2023-03-20 10:30:00)', 'Consultation', 'Completed', 'Ortho consult.', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),

(5, 2, 1, 1, '[2022-11-20 10:00:00, 2022-11-20 11:00:00)', 'Restorative', 'Completed', 'Filling on #30.', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP);

INSERT INTO "plan_items" ("plan_item_id", "plan_id", "code_id", "quantity", "fee_quoted") VALUES

(1, 1, 1, 1, 55.00),

(2, 1, 2, 1, 120.00),

(3, 1, 3, 1, 250.00), -- Plan for filling on #3

(4, 2, 3, 1, 250.00); -- Plan for filling on #30

INSERT INTO "staff_specializations" ("specialization_id", "staff_id", "specialization", "created_at") VALUES

(1, 1, 'Cosmetic Dentistry', CURRENT_TIMESTAMP),

(2, 1, 'Pediatric Dentistry', CURRENT_TIMESTAMP);

INSERT INTO "dentist_procedures" ("procedure_id", "staff_id", "procedure_name", "proficiency_level", "years_experience", "created_at") VALUES

(1, 1, 'Root Canal Therapy', 'Expert', 8, CURRENT_TIMESTAMP),

(2, 1, 'Implants', 'Proficient', 5, CURRENT_TIMESTAMP);

INSERT INTO "admin_responsibilities" ("responsibility_id", "staff_id", "responsibility", "description", "created_at") VALUES

(1, 3, 'Scheduling', 'Manage patient appointment book.', CURRENT_TIMESTAMP),

(2, 3, 'Check-in', 'Greet and check in patients.', CURRENT_TIMESTAMP),

(3, 4, 'Office Management', 'Oversee all administrative staff.', CURRENT_TIMESTAMP);

INSERT INTO "nurse_specializations" ("specialization_id", "staff_id", "specialization", "certification_date", "certification_expiry", "created_at") VALUES

(1, 2, 'Dental Anesthesia', '2022-06-01', '2024-06-01', CURRENT_TIMESTAMP);

-- NOTE: The schema has a design issue. A column ("provider_ref_id")

-- cannot have two foreign keys referencing two different tables.

-- The inserts below *assume* this was intended to be flexible, but

-- the constraints themselves would fail to create in a real DB.

-- I will insert one of each type, assuming the FKs were not applied.

INSERT INTO "healthcare_providers" ("provider_id", "provider_type", "provider_ref_id") VALUES

(1, 'Staff', 1), -- References Dr. Eve Hanson (staff_profiles.staff_id = 1)

(2, 'External', 1); -- References Orthodontics (external_specialists.specialist_id = 1)

INSERT INTO "perio_measurements" ("perio_measurement_id", "perio_exam_id", "tooth_id", "measurement_type", "site", "value", "exam_date") VALUES

(1, 1, 2, 'Probing Depth', 'Mesiobuccal', '3', '2023-01-15'),

(2, 1, 2, 'Probing Depth', 'Buccal', '2', '2023-01-15'),

(3, 1, 2, 'Probing Depth', 'Distobuccal', '3', '2023-01-15'),

(4, 1, 3, 'Probing Depth', 'Mesiobuccal', '3', '2023-01-15'),

(5, 1, 3, 'Probing Depth', 'Buccal', '3', '2023-01-15'),

(6, 1, 3, 'Probing Depth', 'Distobuccal', '3', '2023-01-15');

-- Round 5: Tables dependent on Round 4

INSERT INTO "procedures_log" ("procedure_id", "appointment_id", "patient_id", "staff_id", "code_id", "procedure_date", "fee_charged", "notes") VALUES

(1, 1, 1, 1, 1, '2023-01-15', 55.00, 'Periodic oral eval.'),

(2, 1, 1, 1, 4, '2023-01-15', 30.00, 'PA film x1.'),

(3, 2, 2, 1, 1, '2022-11-10', 55.00, 'Periodic oral eval.'),

(4, 3, 1, 1, 3, '2023-02-01', 250.00, 'Composite filling #3 MOD.'),

(5, 5, 2, 1, 3, '2022-11-20', 250.00, 'Composite filling #30 O.');

INSERT INTO "prescriptions" ("prescription_id", "patient_id", "staff_id", "appointment_id", "drug_name", "dosage", "quantity", "instructions", "refills", "date_prescribed", "created_at") VALUES

(1, 2, 1, 5, 'Amoxicillin', '500mg', 20, '1 tablet 3 times a day for 7 days', 0, '2022-11-20', CURRENT_TIMESTAMP),

(2, 1, 1, 3, 'Ibuprofen', '600mg', 30, '1 tablet as needed for pain, max 4 per day', 1, '2023-02-01', CURRENT_TIMESTAMP);

INSERT INTO "clinical_notes" ("note_id", "appointment_id", "patient_id", "staff_id", "note_title", "note_text", "is_finalized", "created_at", "updated_at") VALUES

(1, 1, 1, 1, 'New Patient Exam', 'Patient presents for initial exam. No immediate concerns. Full mouth x-rays taken. Plan: Prophy and 1 filling.', true, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),

(2, 3, 1, 1, 'Restorative Visit', 'Completed #3 MOD composite. Patient tolerated well. Advised on post-op sensitivity.', true, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP),

(3, 5, 2, 1, 'Restorative Visit', 'Completed #30 O composite. Pre-op infection noted. Prescribed Amoxicillin.', true, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP);

INSERT INTO "documents" ("document_id", "patient_id", "appointment_id", "document_type", "file_name", "storage_identifier", "mime_type", "file_size_bytes", "metadata", "upload_timestamp") VALUES

(1, 1, 1, 'X-Ray', 'fmx_20230115.zip', 's3-bucket/path/fmx_1_abc.zip', 'application/zip', 15728640, '{"series_id": "FMX-123"}', '2023-01-15 09:30:00'),

(2, 1, NULL, 'Consent Form', 'hipaa_consent_alice_smith.pdf', 's3-bucket/path/consent_1_def.pdf', 'application/pdf', 102400, '{"form_version": "v3.1"}', '2023-01-10 08:55:00'),

(3, 2, 2, 'X-Ray', 'bw_20221110.png', 's3-bucket/path/bw_2_ghi.png', 'image/png', 204800, '{"series_id": "BW-456"}', '2022-11-10 14:15:00');

-- Round 6: Tables dependent on Round 5

INSERT INTO "invoices" ("invoice_id", "patient_id", "invoice_date", "due_date", "total_amount", "amount_paid", "status", "created_at") VALUES

(1, 1, '2023-01-15', '2023-02-14', 85.00, 0.00, 'Sent', CURRENT_TIMESTAMP), -- For appointment 1

(2, 2, '2022-11-10', '2022-12-10', 55.00, 55.00, 'Paid', CURRENT_TIMESTAMP), -- For appointment 2

(3, 1, '2023-02-01', '2023-03-03', 250.00, 50.00, 'Partial', CURRENT_TIMESTAMP), -- For appointment 3

(4, 2, '2022-11-20', '2022-12-20', 250.00, 0.00, 'Sent', CURRENT_TIMESTAMP); -- For appointment 5

-- Round 7: Tables dependent on Round 6

INSERT INTO "invoice_items" ("invoice_item_id", "invoice_id", "procedure_id", "amount") VALUES

(1, 1, 1, 55.00), -- Invoice 1: Proc 1 (Eval)

(2, 1, 2, 30.00), -- Invoice 1: Proc 2 (X-Ray)

(3, 2, 3, 55.00), -- Invoice 2: Proc 3 (Eval)

(4, 3, 4, 250.00), -- Invoice 3: Proc 4 (Filling)

(5, 4, 5, 250.00); -- Invoice 4: Proc 5 (Filling)

INSERT INTO "payments" ("payment_id", "invoice_id", "payment_date", "amount", "payment_method", "transaction_id", "created_at") VALUES

(1, 2, '2022-11-15', 55.00, 'Credit Card', 'txn_12345ABC', CURRENT_TIMESTAMP),

(2, 3, '2023-02-05', 50.00, 'Insurance', 'chk_98765', CURRENT_TIMESTAMP);

INSERT INTO "claims" ("claim_id", "invoice_id", "patient_plan_id", "submission_date", "status", "paid_amount", "adjudication_date") VALUES

(1, 1, 1, '2023-01-16', 'Submitted', NULL, NULL),

(2, 2, 2, '2022-11-11', 'Paid', 45.00, '2022-11-25'), -- Insurance paid 45, patient paid 10 (total 55)

(3, 3, 1, '2023-02-02', 'Paid', 50.00, '2023-02-05'),

(4, 4, 2, '2022-11-21', 'Submitted', NULL, NULL);

3. Generate 10 queries based on the tables created in (2). The queries must be based on the following:

i. Group by…having

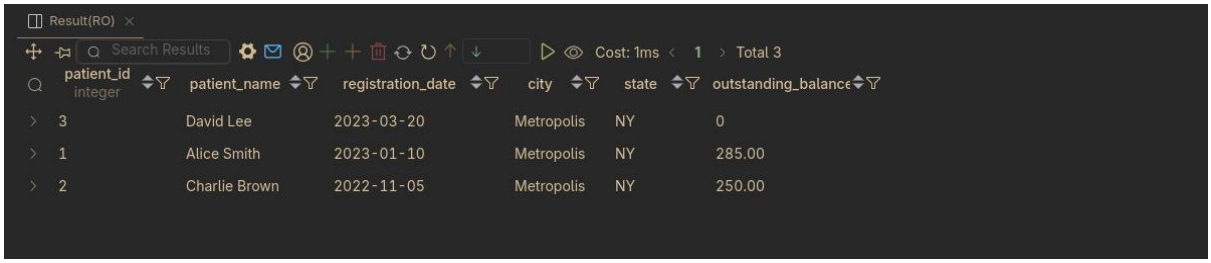
```
-- Use case: Find patients who have had more than 2 appointments and show their total invoice amounts
-- This helps identify high-frequency patients and their financial contribution to the clinic
SELECT
   p.patient_id,
   p.first_name || ' ' || p.last_name AS patient_name,
   COUNT(a.appointment_id) AS total_appointments,
   SUM(i.total_amount) AS total_billed
FROM patients p
JOIN appointments a ON p.patient_id = a.patient_id
LEFT JOIN invoices i ON p.patient_id = i.patient_id
GROUP BY p.patient_id, p.first_name, p.last_name
HAVING COUNT(a.appointment_id) > 2
ORDER BY total_billed DESC;
```

ii. Order by

-- Use case: Generate a comprehensive patient report ordered by registration date and total amount owed

-- This helps the billing department prioritize collection efforts and track patient acquisition patterns

```
SELECT
    p.patient_id,
    p.first_name || ' ' || p.last_name AS patient_name,
    p.registration_date,
    p.city,
    p.state,
    COALESCE(SUM(i.total_amount - i.amount_paid), 0) AS outstanding_balance
FROM patients p
LEFT JOIN invoices i ON p.patient_id = i.patient_id
GROUP BY p.patient_id, p.first_name, p.last_name, p.registration_date, p.city, p.state
ORDER BY p.registration_date DESC, outstanding_balance DESC;
```



iii. Join- Inner , Outer, Self and Natural

a. INNER JOIN Query

-- Use case: Get detailed appointment information with staff and patient details for completed appointments

-- This helps in generating appointment reports and tracking provider productivity

```
SELECT
```

```
    a.appointment_id,
    a.appointment_type,
    a.status,
    p.first_name || ' ' || p.last_name AS patient_name,
    sp.first_name || ' ' || sp.last_name AS staff_name,
    c.clinic_name,
    lower(a.appointment_period) AS appointment_start,
    a.notes
FROM appointments a
INNER JOIN patients p ON a.patient_id = p.patient_id
INNER JOIN staff_profiles sp ON a.staff_id = sp.staff_id
INNER JOIN clinics c ON a.clinic_id = c.clinic_id
WHERE a.status = 'Completed';
```

appointment_id integer	appointment_type	status	patient_name	staff_name	clinic_name	appointment_start	notes
2	Periodic Exam	Completed	Charlie Brown	Eve Hanson	Downtown Dental Care	2022-11-10 14:00:00	6-month checkup.
5	Restorative	Completed	Charlie Brown	Eve Hanson	Downtown Dental Care	2022-11-20 10:00:00	Filling on #30.
1	New Patient Exam	Completed	Alice Smith	Eve Hanson	Downtown Dental Care	2023-01-15 09:00:00	New patient, full exam an
3	Restorative	Completed	Alice Smith	Eve Hanson	Downtown Dental Care	2023-02-01 11:00:00	Filling on #3.
4	Consultation	Completed	David Lee	Eve Hanson	Uptown Smiles	2023-03-20 10:00:00	Ortho consult.

b. LEFT OUTER JOIN Query

```
-- Use case: Show all patients and their emergency contacts (including patients without emergency contacts)
-- This is crucial for patient safety and emergency preparedness
SELECT
    p.patient_id,
    p.first_name || ' ' || p.last_name AS patient_name,
    p.date_of_birth,
    COALESCE(ec.full_name, 'No Emergency Contact') AS emergency_contact_name,
    COALESCE(ec.contact_relationship, 'N/A') AS relationship,
    COALESCE(ec.phone_number, 'No Phone') AS emergency_phone
```

```
FROM patients p
LEFT OUTER JOIN emergency_contacts ec ON p.patient_id = ec.patient_id
ORDER BY p.last_name;
```

c. SELF JOIN Query

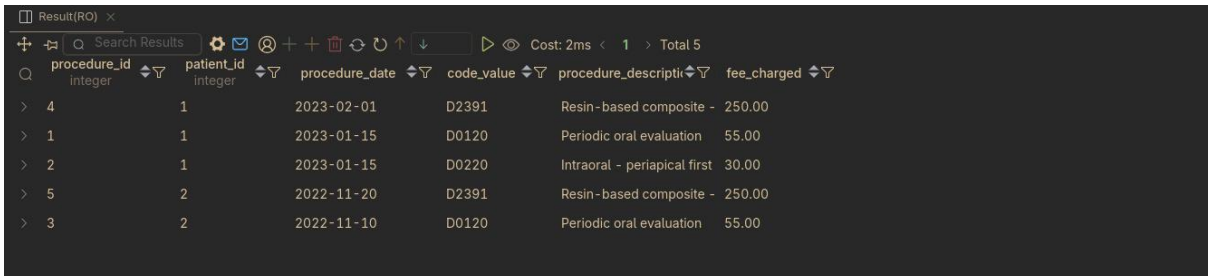
```
-- Use case: Find patients who live in the same city (for potential group appointments or referrals)
-- This helps in community outreach and patient referral programs
SELECT DISTINCT
    p1.patient_id AS patient1_id,
    p1.first_name || ' ' || p1.last_name AS patient1_name,
    p2.patient_id AS patient2_id,
    p2.first_name || ' ' || p2.last_name AS patient2_name,
    p1.city,
    p1.state,
    p1.last_name
FROM patients p1
JOIN patients p2 ON p1.city = p2.city AND p1.state = p2.state AND p1.patient_id < p2.patient_id
ORDER BY p1.city, p1.last_name;
```

d. NATURAL JOIN Query (Modified to work with the schema)

-- Use case: Get patient procedures with treatment codes using common patient_id

-- This helps in treatment analysis and clinical reporting

```
SELECT
    pl.procedure_id,
    pl.patient_id,
    pl.procedure_date,
    tc.code_value,
    tc.description AS procedure_description,
    pl.fee_charged
FROM procedures_log pl, treatment_codes tc, patients p
WHERE pl.code_id = tc.code_id
AND pl.patient_id = p.patient_id
ORDER BY pl.procedure_date DESC;
```



procedure_id integer	patient_id integer	procedure_date	code_value	procedure_descripti	fee_charged
4	1	2023-02-01	D2391	Resin-based composite -	250.00
1	1	2023-01-15	D0120	Periodic oral evaluation	55.00
2	1	2023-01-15	D0220	Intraoral - periapical first	30.00
5	2	2022-11-20	D2391	Resin-based composite -	250.00
3	2	2022-11-10	D0120	Periodic oral evaluation	55.00

iv. Subquery- Independent and Co-related

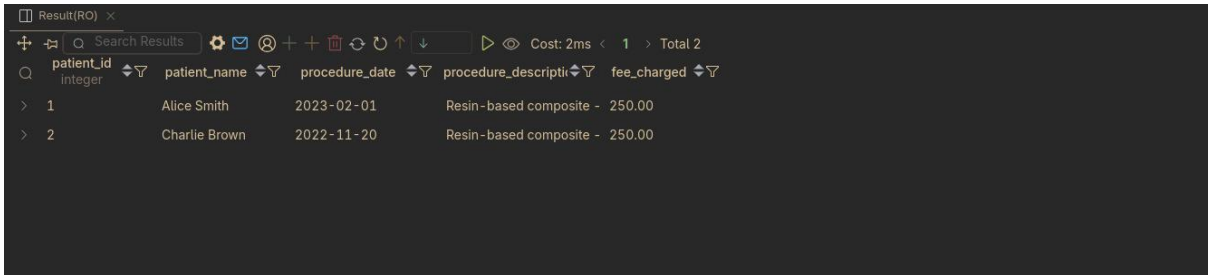
a. INDEPENDENT SUBQUERY Query

-- Use case: Find all patients who have procedures more expensive than the average procedure cost

-- This helps identify high-value treatments and patient spending patterns

```
SELECT
    p.patient_id,
    p.first_name || ' ' || p.last_name AS patient_name,
    pl.procedure_date,
    tc.description AS procedure_description,
    pl.fee_charged
```

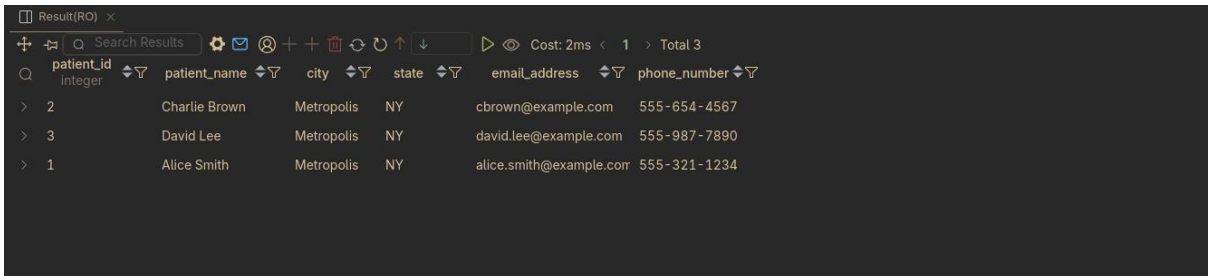
```
FROM patients p
JOIN procedures_log pl ON p.patient_id = pl.patient_id
JOIN treatment_codes tc ON pl.code_id = tc.code_id
WHERE pl.fee_charged > (
    SELECT AVG(fee_charged)
    FROM procedures_log
)
ORDER BY pl.fee_charged DESC;
```



b. CORRELATED SUBQUERY Query

```
-- Use case: Find patients whose most recent appointment was more than 6 months ago (due for checkup)
-- This helps in patient recall and preventive care scheduling
SELECT
    p.patient_id,
    p.first_name || ' ' || p.last_name AS patient_name,
    p.city,
    p.state,
    ppe.email_address,
    ppn.phone_number
FROM patients p
LEFT JOIN patient_email_addresses ppe ON p.patient_id = ppe.patient_id AND ppe.is_primary = true
LEFT JOIN patient_phone_numbers ppn ON p.patient_id = ppn.patient_id AND ppn.is_primary = true
WHERE (
    SELECT MAX(lower(appointment_period))
```

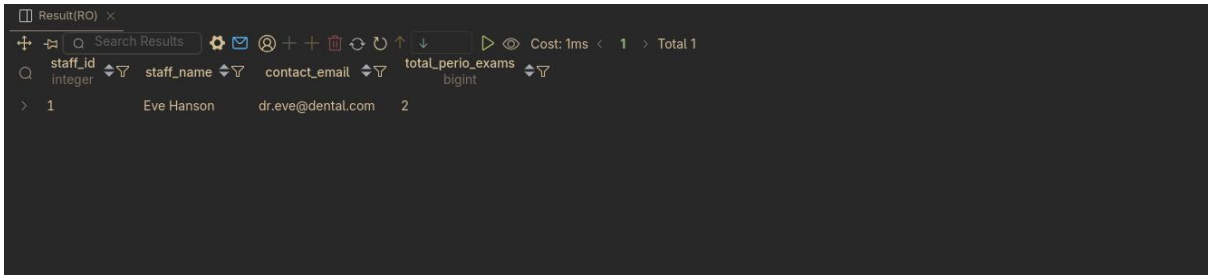
```
    FROM appointments a
    WHERE a.patient_id = p.patient_id
) < CURRENT_DATE - INTERVAL '6 months'
OR NOT EXISTS (
    SELECT 1
    FROM appointments a
    WHERE a.patient_id = p.patient_id
)
ORDER BY p.last_name;
```



v. Exists and Not exists

a. EXISTS Query

```
-- Use case: Find all staff members who have conducted periodontal examinations
-- This helps identify staff expertise and workload distribution
SELECT
    sp.staff_id,
    sp.first_name || ' ' || sp.last_name AS staff_name,
    sp.contact_email,
    COUNT(pe.perio_exam_id) AS total_perio_exams
FROM staff_profiles sp
JOIN perio_exams pe ON sp.staff_id = pe.staff_id
GROUP BY sp.staff_id, sp.first_name, sp.last_name, sp.contact_email
ORDER BY total_perio_exams DESC;
```

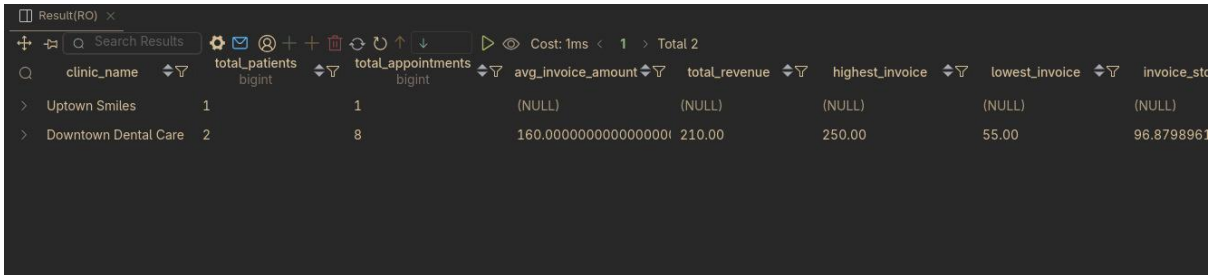


b. NOT EXISTS Query

```
-- Use case: Find patients who have never had any invoices (potential billing issues or new patients)
-- This helps identify billing gaps and track new patient onboarding
SELECT
    p.patient_id,
    p.first_name || ' ' || p.last_name AS patient_name,
    p.registration_date,
    p.city,
    p.state,
    COUNT(a.appointment_id) AS total_appointments
FROM patients p
LEFT JOIN appointments a ON p.patient_id = a.patient_id
WHERE NOT EXISTS (
    SELECT 1
    FROM invoices i
    WHERE i.patient_id = p.patient_id
)
GROUP BY p.patient_id, p.first_name, p.last_name, p.registration_date, p.city, p.state
ORDER BY p.registration_date;
```

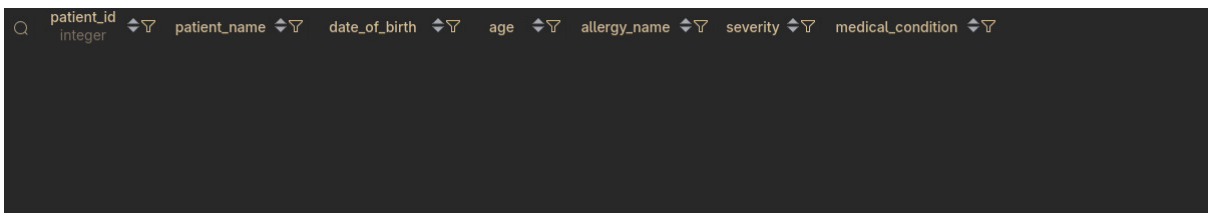
vi. Aggregate functions

```
-- Use case: Comprehensive clinic performance dashboard showing key metrics
-- This provides management with essential KPIs for business decision making
SELECT
    c.clinic_name,
    COUNT(DISTINCT p.patient_id) AS total_patients,
    COUNT(a.appointment_id) AS total_appointments,
    AVG(i.total_amount) AS avg_invoice_amount,
    SUM(i.amount_paid) AS total_revenue,
    MAX(i.total_amount) AS highest_invoice,
    MIN(i.total_amount) AS lowest_invoice,
    STDDEV(i.total_amount) AS invoice_std_dev
FROM clinics c
LEFT JOIN appointments a ON c.clinic_id = a.clinic_id
LEFT JOIN patients p ON a.patient_id = p.patient_id
LEFT JOIN invoices i ON p.patient_id = i.patient_id
GROUP BY c.clinic_id, c.clinic_name
ORDER BY total_revenue DESC;
```



vii. Query having Boolean operators

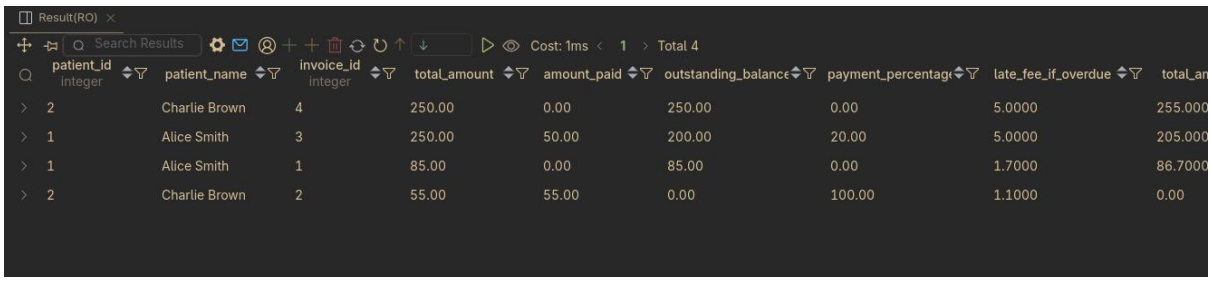
```
-- Use case: Find high-risk patients (seniors with allergies and active medical conditions)
-- This helps in treatment planning and risk assessment
SELECT
    p.patient_id,
    p.first_name || ' ' || p.last_name AS patient_name,
    p.date_of_birth,
    EXTRACT(YEAR FROM AGE(CURRENT_DATE, p.date_of_birth)) AS age,
    pa.allergy_name,
    pa.severity,
    mh.description AS medical_condition
FROM patients p
JOIN patient_allergies pa ON p.patient_id = pa.patient_id
JOIN medical_history mh ON p.patient_id = mh.patient_id
WHERE (EXTRACT(YEAR FROM AGE(CURRENT_DATE, p.date_of_birth)) >= 65)
AND (pa.severity = 'Severe' OR pa.severity = 'Moderate')
AND mh.is_active = true
AND NOT (mh.history_type = 'Surgery' AND mh.end_date IS NOT NULL)
ORDER BY age DESC, pa.severity DESC;
```



Note: No output

viii. Query having arithmetic operators

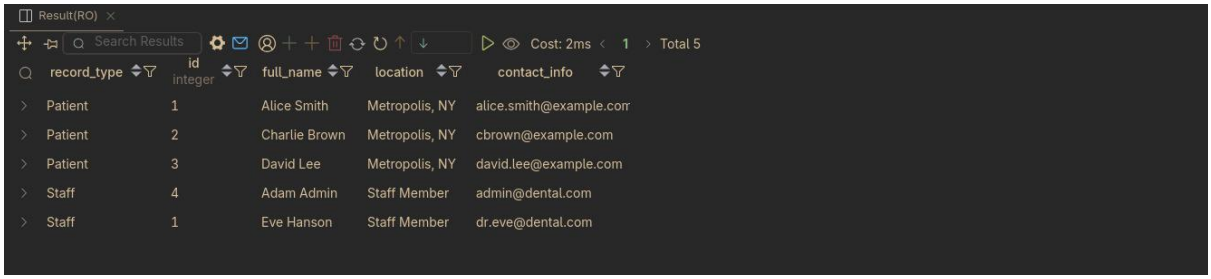
```
-- Use case: Calculate payment efficiency and outstanding balances for financial reporting
-- This helps the finance team track collection rates and cash flow
SELECT
    p.patient_id,
    p.first_name || ' ' || p.last_name AS patient_name,
    i.invoice_id,
    i.total_amount,
    i.amount_paid,
    (i.total_amount - i.amount_paid) AS outstanding_balance,
    ROUND(((i.amount_paid / i.total_amount) * 100), 2) AS payment_percentage,
    (i.total_amount * 0.02) AS late_fee_if_overdue,
    CASE
        WHEN i.due_date < CURRENT_DATE AND (i.total_amount - i.amount_paid) > 0
        THEN (i.total_amount - i.amount_paid) + (i.total_amount * 0.02)
        ELSE (i.total_amount - i.amount_paid)
    END AS total_amount_due
FROM patients p
JOIN invoices i ON p.patient_id = i.patient_id
WHERE i.total_amount > 0
ORDER BY outstanding_balance DESC;
```



patient_id integer	patient_name	invoice_id integer	total_amount	amount_paid	outstanding_balance	payment_percentage	late_fee_if_overdue	total_an
2	Charlie Brown	4	250.00	0.00	250.00	0.00	5.0000	255.000
1	Alice Smith	3	250.00	50.00	200.00	20.00	5.0000	205.000
1	Alice Smith	1	85.00	0.00	85.00	0.00	1.7000	86.7000
2	Charlie Brown	2	55.00	55.00	0.00	100.00	1.1000	0.00

ix. A search query using string operators

```
-- Use case: Search for patients and staff with flexible name matching for reception lookup
-- This helps reception staff quickly find records during patient check-in
SELECT
    'Patient' AS record_type,
    p.patient_id AS id,
    p.first_name || ' ' || p.last_name AS full_name,
    p.city || ', ' || p.state AS location,
    COALESCE(ppe.email_address, 'No Email') AS contact_info
FROM patients p
LEFT JOIN patient_email_addresses ppe ON p.patient_id = ppe.patient_id AND ppe.is_primary = true
WHERE UPPER(p.first_name) LIKE '%A%'
   OR UPPER(p.last_name) SIMILAR TO '%(SMITH|BROWN|LEE)%'
   OR p.first_name ILIKE 'ch%'

UNION ALL

SELECT
    'Staff' AS record_type,
    sp.staff_id AS id,
    sp.first_name || ' ' || sp.last_name AS full_name,
    'Staff Member' AS location,
    sp.contact_email AS contact_info
FROM staff_profiles sp
WHERE sp.first_name ~ '^[AE]'
   OR POSITION('ob' IN LOWER(sp.last_name)) > 0
ORDER BY record_type, full_name;
```



x. Usage of to_char, extract

```
-- Use case: Generate monthly appointment and revenue reports for business analysis
-- This provides detailed temporal analysis for scheduling optimization and financial planning
SELECT
    TO_CHAR(lower(a.appointment_period), 'YYYY-MM') AS year_month,
    TO_CHAR(lower(a.appointment_period), 'Month') AS month_name,
    EXTRACT(QUARTER FROM lower(a.appointment_period)) AS quarter,
    EXTRACT(DOW FROM lower(a.appointment_period)) AS day_of_week,
    CASE EXTRACT(DOW FROM lower(a.appointment_period))
        WHEN 0 THEN 'Sunday'
        WHEN 1 THEN 'Monday'
        WHEN 2 THEN 'Tuesday'
        WHEN 3 THEN 'Wednesday'
        WHEN 4 THEN 'Thursday'
        WHEN 5 THEN 'Friday'
        WHEN 6 THEN 'Saturday'
    END AS day_name,
    COUNT(*) AS appointment_count,
    SUM(COALESCE(pl.fee_charged, 0)) AS total_revenue
FROM appointments a
LEFT JOIN procedures_log pl ON a.appointment_id = pl.appointment_id
WHERE lower(a.appointment_period) >= '2022-01-01'
GROUP BY
```

```
    TO_CHAR(lower(a.appointment_period), 'YYYY-MM'),
    TO_CHAR(lower(a.appointment_period), 'Month'),
    EXTRACT(QUARTER FROM lower(a.appointment_period)),
    EXTRACT(DOW FROM lower(a.appointment_period))
ORDER BY year_month, day_of_week;
```

year_month	month_name	quarter	day_of_week	day_name	appointment_count (bigint)	total_revenue
2022-11	November	4	0	Sunday	1	250.00
2022-11	November	4	4	Thursday	1	55.00
2023-01	January	1	0	Sunday	2	85.00
2023-02	February	1	3	Wednesday	1	250.00
2023-03	March	1	1	Monday	1	0

xi. Between, IN, Not between, Not IN

```
-- Use case: Analyze treatment patterns and identify patients needing specific follow-up care
-- This helps in clinical workflow management and treatment outcome tracking
SELECT
    p.patient_id,
    p.first_name || ' ' || p.last_name AS patient_name,
    tc.description AS procedure_description,
    pl.procedure_date,
    pl.fee_charged,
    CASE
        WHEN tc.code_value IN ('D0120', 'D1110') THEN 'Preventive Care'
        WHEN tc.code_value IN ('D2391') THEN 'Restorative Care'
        WHEN tc.code_value IN ('D0220') THEN 'Diagnostic'
        ELSE 'Other'
    END AS procedure_category
FROM patients p
JOIN procedures_log pl ON p.patient_id = pl.patient_id
JOIN treatment_codes tc ON pl.code_id = tc.code_id
```

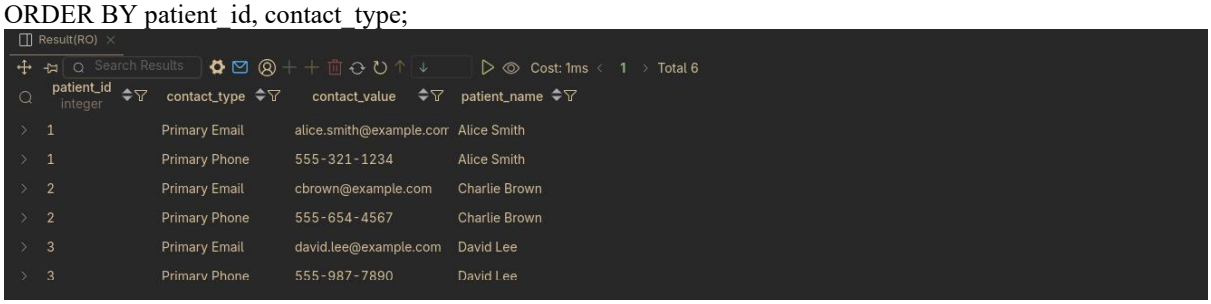
```
WHERE pl.procedure_date BETWEEN '2022-01-01' AND '2023-12-31'

AND pl.fee_charged BETWEEN 50.00 AND 300.00

AND tc.code_value NOT IN ('D0270', 'D0330')  -- Excluding specific codes

AND p.patient_id NOT BETWEEN 100 AND 200     -- Excluding specific patient range

AND pl.procedure_date NOT BETWEEN '2023-06-01' AND '2023-06-30'  -- Excluding June 2023

ORDER BY pl.procedure_date DESC, p.last_name;
```

patient_id integer	patient_name	procedure_descripti	procedure_date	fee_charged	procedure_category
1	Alice Smith	Resin-based composite -	2023-02-01	250.00	Restorative Care
1	Alice Smith	Periodic oral evaluation	2023-01-15	55.00	Preventive Care
2	Charlie Brown	Resin-based composite -	2022-11-20	250.00	Restorative Care
2	Charlie Brown	Periodic oral evaluation	2022-11-10	55.00	Preventive Care

xii. Set operations

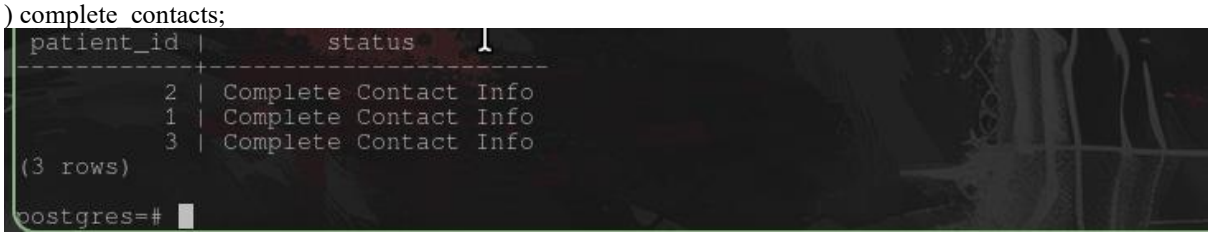
```
-- Use case: Comprehensive contact management for patient communication campaigns

-- This helps marketing and patient care teams coordinate outreach efforts


-- UNION: All patient contacts for emergency notifications

(SELECT

   p.patient_id,

   'Primary Email' AS contact_type,

   ppe.email_address AS contact_value,

   p.first_name || ' ' || p.last_name AS patient_name

FROM patients p

JOIN patient_email_addresses ppe ON p.patient_id = ppe.patient_id

WHERE ppe.is_primary = true)


UNION


(SELECT
```

```
    p.patient_id,
    'Primary Phone' AS contact_type,
    ppn.phone_number AS contact_value,
    p.first_name || ' ' || p.last_name AS patient_name
FROM patients p
JOIN patient_phone_numbers ppn ON p.patient_id = ppn.patient_id
WHERE ppn.is_primary = true)

ORDER BY patient_id, contact_type;
```




```
-- Additional SET OPERATION Examples:

-- INTERSECT: Find patients who have both email and phone contacts (complete contact info)
SELECT patient_id, 'Complete Contact Info' AS status
FROM (
    SELECT DISTINCT p.patient_id
    FROM patients p
    JOIN patient_email_addresses ppe ON p.patient_id = ppe.patient_id
    WHERE ppe.is_primary = true

    INTERSECT

    SELECT DISTINCT p.patient_id
    FROM patients p
```

```
    JOIN patient_phone_numbers ppn ON p.patient_id = ppn.patient_id
    WHERE ppn.is_primary = true
) complete_contacts;
```




```
-- EXCEPT: Find patients with appointments but no invoices (potential billing issues)
SELECT DISTINCT
    a.patient_id,
    'Missing Invoice' AS billing_status
FROM appointments a
WHERE a.status = 'Completed'

EXCEPT

SELECT DISTINCT
    i.patient_id,
    'Has Invoice' AS billing_status
FROM invoices i;
```