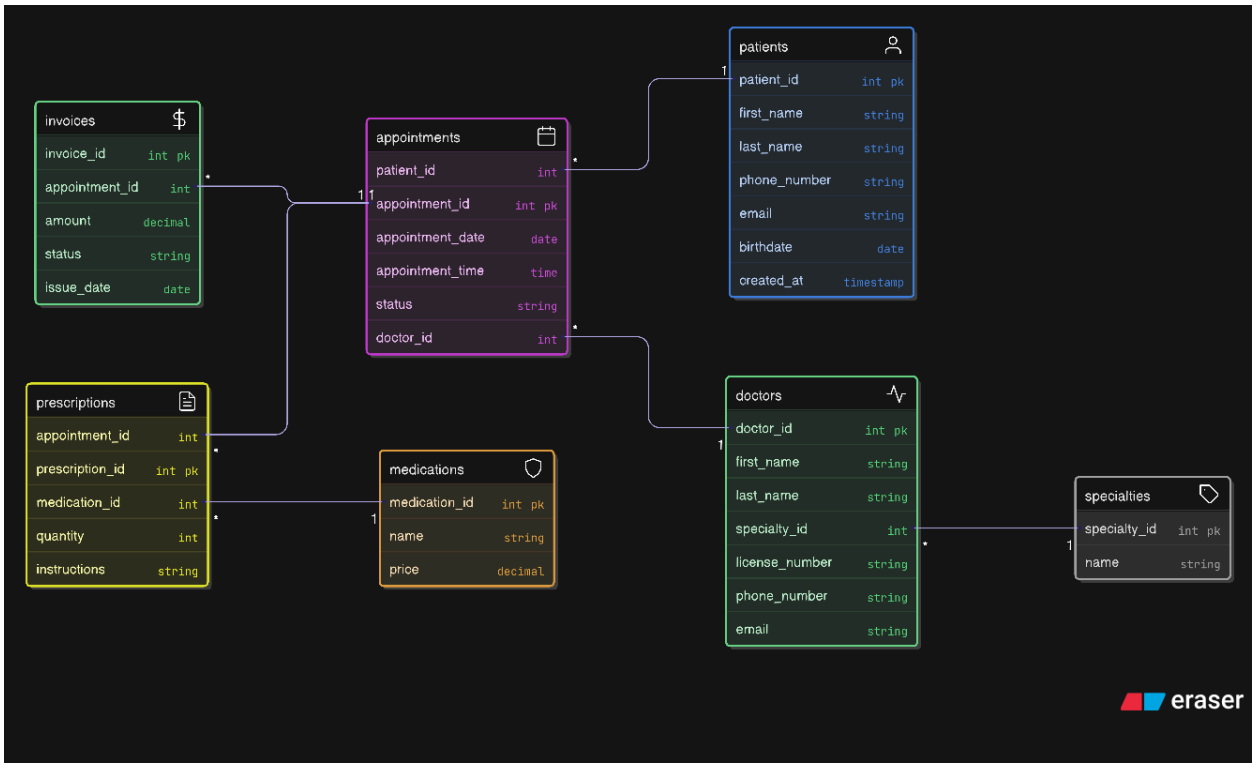
Clinic Appointment Management System - Project Document

1. Brief Description of Your Project

This project involves the design and implementation of a comprehensive database system for managing a medical clinic. The system facilitates the storage and management of information related to patients, doctors, appointments, medications, prescriptions, invoices, and doctor specialties. It includes functionalities for data entry, retrieval, analysis, and updates, along with advanced database features like views, sequences, and indexes to ensure efficient and robust operation.

The primary objective is to streamline clinic operations by providing a centralized and organized system for appointment scheduling, patient record keeping, prescription management, and billing.

The schema visualization:



2. List of All Tables and Their Purpose

Here's a list of the tables created for the Clinic Appointment Management System, along with their respective purposes:

- **Specialties**:
  - **Purpose**: Defines and stores various medical specialties (e.g., General Practice, Pediatrics, Cardiology). This table ensures consistency and proper categorization of doctors.
  - **Key Fields**: specialty_id (Primary Key), name (Unique specialty name).
- **Patients**:
  - **Purpose**: Stores detailed information about each patient registered with the clinic.
  - **Key Fields**: patient_id (Primary Key), first_name, last_name, phone_number (Unique), email (Unique), birthdate, created_at.
- **Doctors**:
  - **Purpose**: Stores information about the doctors working at the clinic, including their personal details, specialty, and license.
  - **Key Fields**: doctor_id (Primary Key), first_name, last_name, specialty_id (Foreign Key referencing Specialties), license_number (Unique), phone_number (Unique), email (Unique).
- **Appointments**:
  - **Purpose**: Manages the scheduling of appointments between patients and doctors. It records the date, time, and status of each appointment.
  - **Key Fields**: appointment_id (Primary Key), patient_id (Foreign Key referencing Patients), doctor_id (Foreign Key referencing Doctors), appointment_date, appointment_time, status (e.g., 'Scheduled', 'Completed', 'Cancelled', 'No-Show').
- **Medications**:
  - **Purpose**: Stores details about the various medications available or prescribed by the clinic.
  - **Key Fields**: medication_id (Primary Key), name (Unique medication name), price.
- **Prescriptions**:
  - **Purpose**: Records the prescriptions issued to patients during their appointments, linking specific medications to appointments.
  - **Key Fields**: prescription_id (Primary Key), appointment_id (Foreign Key referencing Appointments), medication_id (Foreign Key referencing

Medications), quantity, instructions.
- **Invoices**:
  - **Purpose**: Stores billing information related to appointments, including the amount due and payment status.
  - **Key Fields**: invoice_id (Primary Key), appointment_id (Unique Foreign Key referencing Appointments), amount, status (e.g., 'Pending', 'Paid', 'Cancelled', 'Refunded'), issue_date.

3. Summary of the SQL Queries Included

The SQL queries cover a wide range of operations, from basic data retrieval to complex analytical queries and data management features.

3.1. Database Design and Table Creation

- CREATE TABLE statements for all seven tables (Specialties, Patients, Doctors, Appointments, Medications, Prescriptions, Invoices).
- Definition of PRIMARY KEY, FOREIGN KEY relationships, and constraints like NOT NULL, UNIQUE, and CHECK for data integrity.
- COMMENT ON TABLE and COMMENT ON COLUMN for documentation.

3.2. Data Operations (DML)

- INSERT INTO statements to populate all tables with realistic sample data (minimum 5 rows per table).
- Specific operations demonstrated:
  - Adding a new patient.
  - Adding a new appointment.
  - Adding a new invoice (using a DODECLARE...END ; block for dynamic appointment_id retrieval).
  - Updating a patient's phone number.
  - Modifying a doctor's specialty.
  - Deleting a cancelled appointment.
- Demonstration of COMMIT and ROLLBACK for transaction control, showing how changes can be permanently saved or undone.

3.3. Data Retrieval and Queries (DQL)

- **Basic Queries**:
  - Listing all patients.
  - Listing all doctors with their specialties (using JOIN).
  - Displaying all appointments with patient and doctor names.
- **Filtered and Sorted Queries**:

    - Showing appointments on a specific date (using EXTRACT).
    - Listing patients registered after a specific year (using EXTRACT on birthdate).
    - Sorting doctors by name and specialty (using ORDER BY).
- **Queries Using Functions**:
    - Displaying patient names in uppercase (using UPPER()).
    - Calculating patient age from birthdate (using AGE()).
    - Formatting appointment date clearly (implicit formatting through SELECT and JOINs).
- **Analytical Queries**:
    - Counting appointments per doctor (using COUNT() and GROUP BY).
    - Calculating total payments per patient (using SUM() and GROUP BY).
    - Counting total registered patients (using COUNT(*)).
- **Multi-table Queries**:
    - Showing appointment details including doctor and patient names (using multiple JOINs).
    - Listing prescriptions with patient and medication names (using multiple JOINs).
- **Advanced Queries**:
    - Showing patients with more than 3 appointments (using GROUP BY and HAVING).
    - Showing doctors with no appointments this week (using LEFT JOIN, GROUP BY, and HAVING).
    - Showing patients who visited more than one doctor and received prescriptions (using JOINs, COUNT(DISTINCT), GROUP BY, and HAVING).
    - Showing patients who never received a prescription (using LEFT JOIN, GROUP BY, and HAVING).

3.4. Data Management Features

- **View**:
    - Creation of a VIEW named today to display today's appointments with doctor and patient names for easy access.
    - SELECT * FROM today; to demonstrate its usage.
- **Sequence**:
    - Creation of a SEQUENCE named seq to auto-generate values, demonstrating its use for potential invoice numbers or patient IDs.
    - SELECT NEXTVAL('seq'); to show sequence increment.
- **Index**:
    - Creation of an INDEX named d_name on the first_name column of the Doctors table to improve query performance on doctor names.

- EXPLAIN ANALYZE statement to illustrate the performance before and after index creation (though the output of EXPLAIN ANALYZE itself is not included in this document, the command is present in the SQL file).