**HOSPITAL MANAGEMENT SYSTEM**

Abstract:

* This database management system is designed to handle the information of doctors and patients within a hospital.
* It provides an organized structure for storing details such as names, addresses, contact numbers, email addresses, and other essential information.
* The system consists of two main tables: "doctors" and "patients.
* " The "doctors" table holds data related to healthcare professionals working at the hospital, while the "patients" table stores information about the patients receiving medical care.

Aim:

* The aim of this database is to efficiently manage and organize information about doctors and patients within a hospital setting.
* It allows healthcare professionals to access patient records and facilitates appointments, ensuring seamless communication between doctors and their patients.
* By maintaining accurate and up-to-date data, the hospital can provide better healthcare services and improve overall patient care.

Introduction:

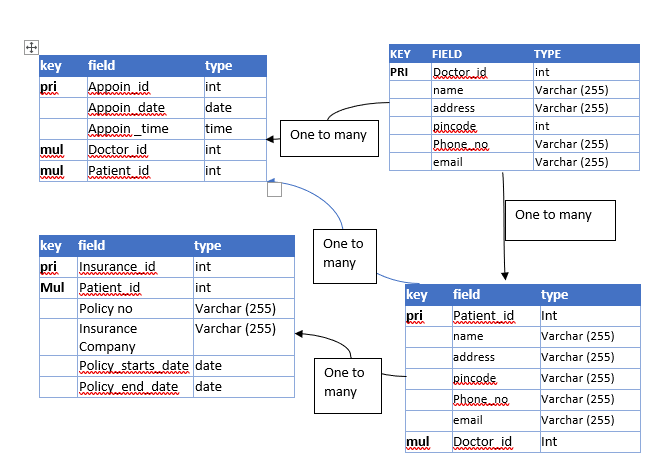
* In the rapidly evolving healthcare industry, efficient management of doctor and patient information is crucial for providing high-quality medical services.
* This database system is designed to address the challenges associated with storing and retrieving data related to doctors and patients.
* It ensures the smooth operation of the hospital by maintaining detailed records and simplifying the process of appointment scheduling.
* The "doctors" table contains essential details about healthcare providers, including their names, addresses, contact numbers, and email addresses.
* Each doctor is assigned a unique doctor\_id as the primary key, ensuring data integrity and easy identification.
* The "patients" table stores information about patients, including their names, addresses, pin codes, phone numbers, and email addresses.
* Each patient record is associated with a doctor through the doctor\_id, enabling efficient tracking of doctor-patient relationships.

Objective:

* ***Efficient Data Storage:*** To store and organize doctor and patient information in a structured and efficient manner, reducing the risk of data loss or inconsistency.
* ***Simplified Doctor***-Patient Interaction: To facilitate communication between doctors and patients by maintaining accurate contact information and doctor-patient associations.
* ***Appointment Management:*** To streamline the appointment scheduling process, making it easier for patients to access medical care and for doctors to manage their schedules.
* ***Data Integrity:*** To ensure data accuracy and integrity by using primary keys and constraints, minimizing errors in the system.

**ER DIAGRAM**

HOSPITAL MANAGMENT SYSTEM



Structure of tables:

1)Doctor:

//creating a doctor table

create table doctor(

doctor\_id int,

name varchar(255) not null,

address varchar(255) not null,

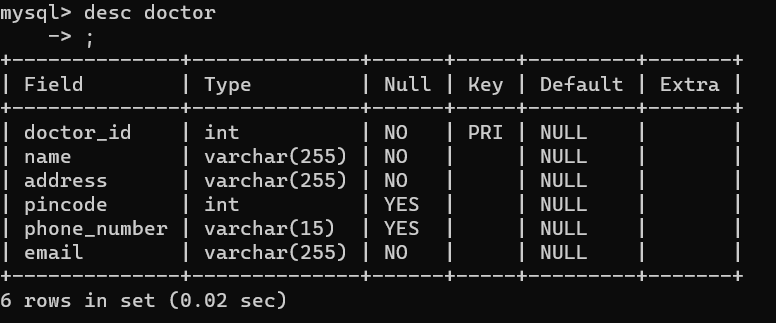
pincode int,

phone\_number int not null,

email varchar(255) not null,

primary key (doctor\_id)

);



//inserting data of doctors

INSERT INTO doctor (doctor\_id,name, address,pincode,phone\_number, email)

VALUES

(1,'Dr. yash', 'mumbai', '421201','97853421423','dryash@gmail.com'),

(2,'Dr. dhairya', 'thane', '42701', '9891234568', 'drdhariya@gmail.com'),

(3,'Dr. ram', 'thane', '42701','9981234569','drram@gmail.com'),

(4,'Dr. sham','dombivli','42701','9091234570','drsham@gmail.com'),

(5,'Dr. sneha ', 'mumbai','42701','9001234571','sneha@gmail.com'),

(6,'Dr. rutu','','42701', '9971234572', 'drrutu@gmail.com'),

(7,'Dr. avantika','kurla','42701','9761234573','dravantika@gmail.com'),

(8,'Dr. shruti', 'parel', '42701', '9876564574','drshruti@gmail.com'),

(9,'Dr. anviksha', 'diva', '42701', '9451234575','dranviksha@gmail.com'),

(10,'Dr. ritika', 'badlapur', '42701', '9671234576', 'drritika@gmail.com');



2)Patient:

CREATE TABLE patients (

patient\_id INT ,

name VARCHAR(255) NOT NULL,

address VARCHAR(255) NOT NULL,

pincode VARCHAR(10) NOT NULL,

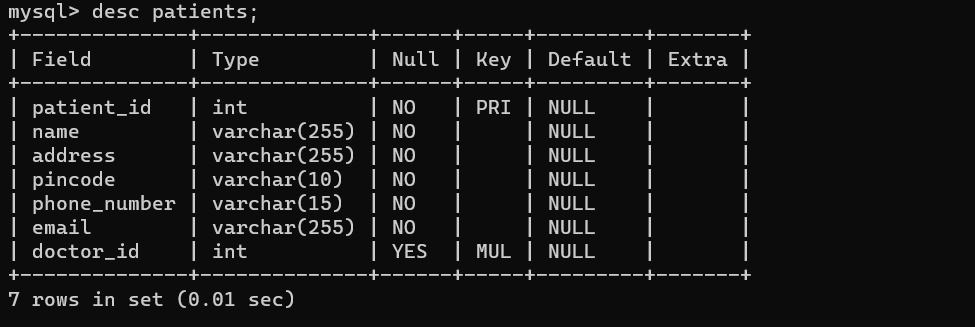
phone\_number VARCHAR(15) NOT NULL,

email VARCHAR(255) NOT NULL,

FOREIGN KEY (doctor\_id) REFERENCES doctors(doctor\_id),

PRIMARY KEY (patient\_id)

);



//inserting the data of patient

INSERT INTO patients (patient\_id, name, address, pincode, phone\_number, email, doctor\_id)

VALUES

(1, 'Rajesh ', 'Mumbai', '400001', '9876543210', 'rajesh@gmail.com', 1),

(2, 'Priya ', 'Delhi', '110001', '9876543211', 'priya@gmail.com', 2),

(3, 'Amit ', 'Bangalore', '560001', '9876543212', 'amit@gmail.com', 2),

(4, 'Sneha', 'Kolkata', '700001', '9876543213', 'sneha@gmail.com', 3),

(5, 'Rahul ', 'Chennai', '600001', '9876543214', 'rahul@gmail.com', 4),

(6, 'Neha ', 'Hyderabad', '500001', '9876543215', 'neha@gmail.com', 5),

(7, 'Ankit', 'Pune', '411001', '9876543216', 'ankit@gmail.com', 5),

(8, 'Pooja', 'Jaipur', '302001', '9876543217', 'pooja@gmail.com', 6),

(9, 'Sanjay', 'Lucknow', '226001', '9876543218', 'sanjay@gmail.com', 6),

(10, 'Suman', 'Ahmedabad', '380001', '9876543219', 'suman@gmail.com', 7),

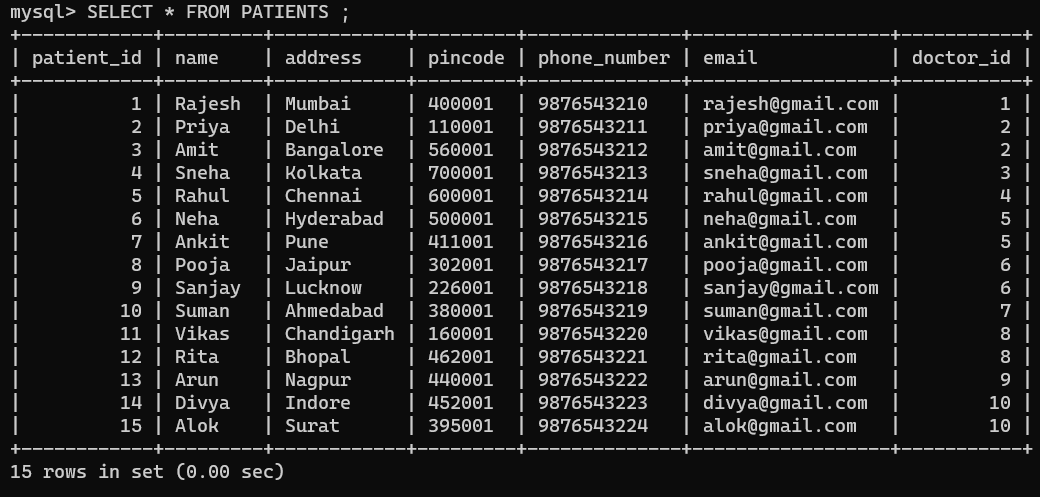
(11, 'Vikas', 'Chandigarh', '160001', '9876543220', 'vikas@gmail.com', 8),

(12, 'Rita ', 'Bhopal', '462001', '9876543221', 'rita@gmail.com', 8),

(13, 'Arun ', 'Nagpur', '440001', '9876543222', 'arun@gmail.com', 9),

(14, 'Divya', 'Indore', '452001', '9876543223', 'divya@gmail.com', 10),

(15, 'Alok ', 'Surat', '395001', '9876543224', 'alok@gmail.com', 10);



3)Appointments:

//creating appoinment table

CREATE TABLE appointments (

appointment\_id INT AUTO\_INCREMENT PRIMARY KEY,

appointment\_date DATE NOT NULL,

appointment\_time TIME NOT NULL,

doctor\_id INT NOT NULL,

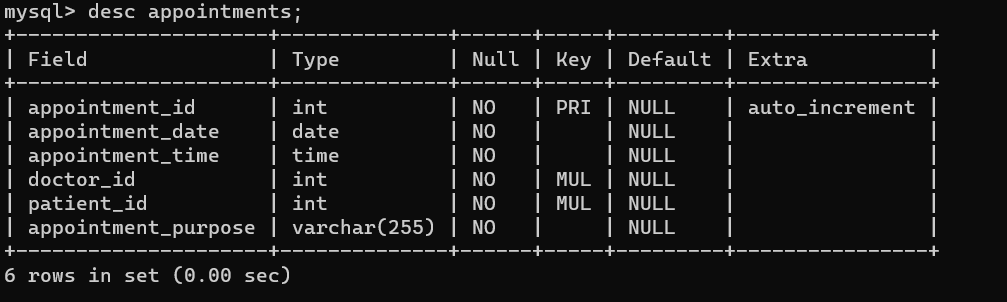
patient\_id INT NOT NULL,

appointment\_purpose VARCHAR(255) NOT NULL,

appointment\_notes TEXT,

FOREIGN KEY (doctor\_id) REFERENCES doctors(doctor\_id),

FOREIGN KEY (patient\_id) REFERENCES patients(patient\_id);



//inserting data in appoinment

INSERT INTO appointments (appointment\_date, appointment\_time, doctor\_id, patient\_id, appointment\_purpose)

VALUES

('2023-10-10', '09:00:00', 1, 1, 'General checkup'),

('2023-10-12', '14:30:00', 2, 2, 'Dental examination'),

('2023-10-15', '11:15:00', 3, 3, 'Cardiology consultatIon'),

('2023-10-16', '15:45:00', 4, 4, 'Orthopedic appointment' ),

('2023-10-18', '10:30:00', 5, 5, 'Eye examination'),

('2023-10-20', '13:00:00', 6, 6, 'ENT checkup' ),

('2023-10-22', '12:45:00', 7, 7, 'Gastroenterology consultation'),

('2023-10-25', '16:30:00', 8, 8, 'Dermatology appointment'),

('2023-10-28', '09:30:00', 9, 9, 'Neurology consultation'),

('2023-10-30', '14:15:00', 10, 10, 'Pediatric checkup'),

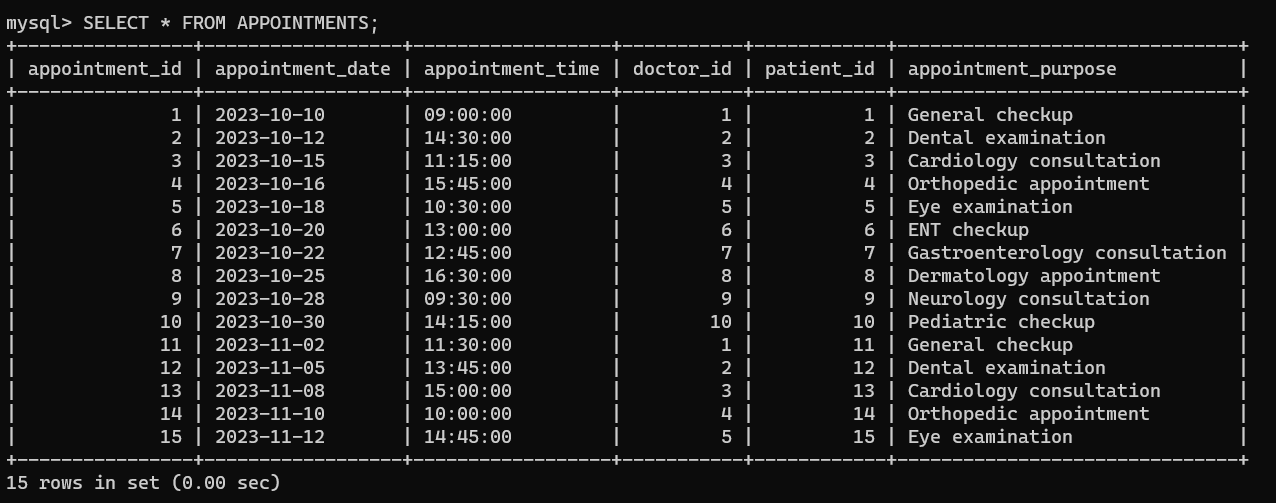
('2023-11-02', '11:30:00', 1, 11, 'General checkup'),

('2023-11-05', '13:45:00', 2, 12, 'Dental examination'),

('2023-11-08', '15:00:00', 3, 13, 'Cardiology consultation'),

('2023-11-10', '10:00:00', 4, 14, 'Orthopedic appointment'),

('2023-11-12', '14:45:00', 5, 15, 'Eye examination');



4)Insurance\_information:

//Create the Insurance Information table

CREATE TABLE insurance\_information (

insurance\_id INT AUTO\_INCREMENT PRIMARY KEY,

patient\_id INT NOT NULL,

policy\_number VARCHAR(20) NOT NULL,

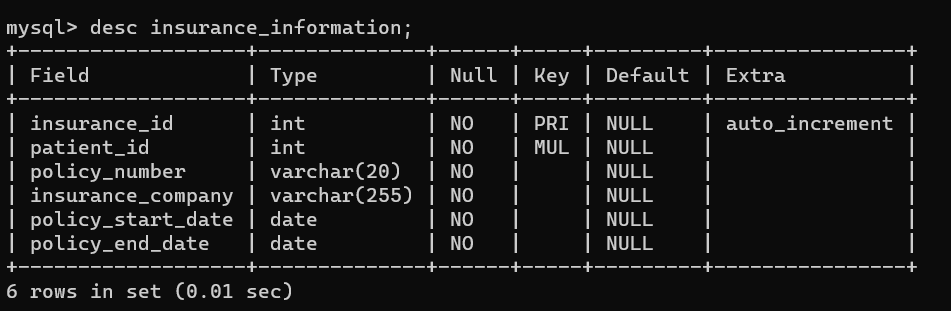
insurance\_company VARCHAR(255) NOT NULL,

policy\_start\_date DATE NOT NULL,

policy\_end\_date DATE NOT NULL,

FOREIGN KEY (patient\_id) REFERENCES patients(patient\_id)

);



//Insert data into the Insurance Information table

INSERT INTO insurance\_information (patient\_id, policy\_number, insurance\_company, policy\_start\_date, policy\_end\_date)

VALUES

(1, 'POL123456', 'A Insurance', '2023-01-01', '2024-01-01'),

(2, 'POL789012', 'B Insurance', '2023-02-01', '2024-02-01'),

(3, 'POL345678', 'C Insurance', '2023-03-01', '2024-03-01'),

(4, 'POL901234', 'D Insurance', '2023-04-01', '2024-04-01'),

(5, 'POL567890', 'A Insurance', '2023-05-01', '2024-05-01'),

(6, 'POL234567', 'C Insurance', '2023-06-01', '2024-06-01'),

(7, 'POL890123', 'B Insurance', '2023-07-01', '2024-07-01'),

(8, 'POL456789', 'D Insurance', '2023-08-01', '2024-08-01'),

(9, 'POL012345', 'C Insurance', '2023-09-01', '2024-09-01'),

(10, 'POL678901', 'A Insurance','2023-10-01', '2024-10-01'),

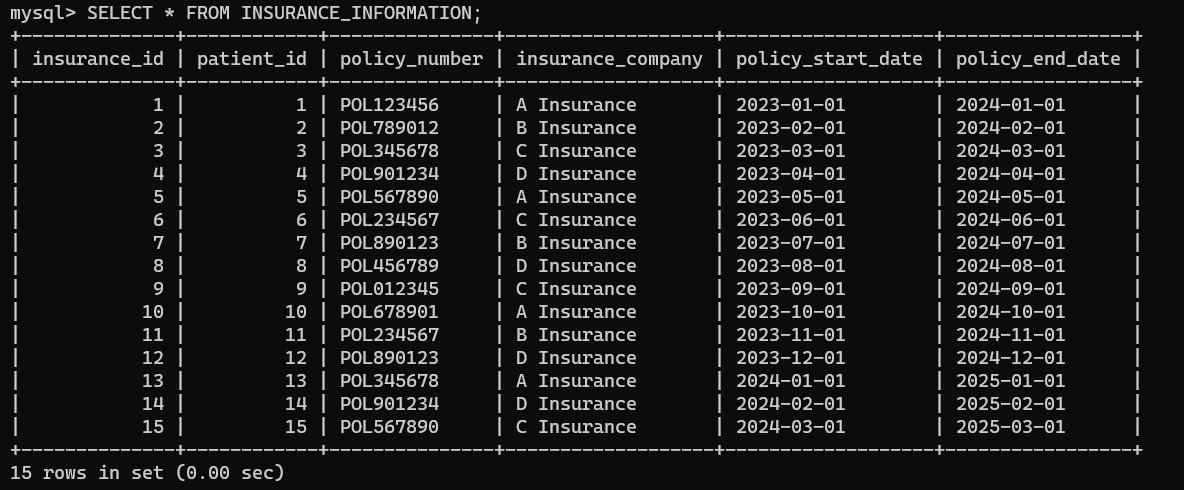
(11, 'POL234567', 'B Insurance','2023-11-01', '2024-11-01'),

(12, 'POL890123', 'D Insurance','2023-12-01', '2024-12-01'),

(13, 'POL345678', 'A Insurance','2024-01-01', '2025-01-01'),

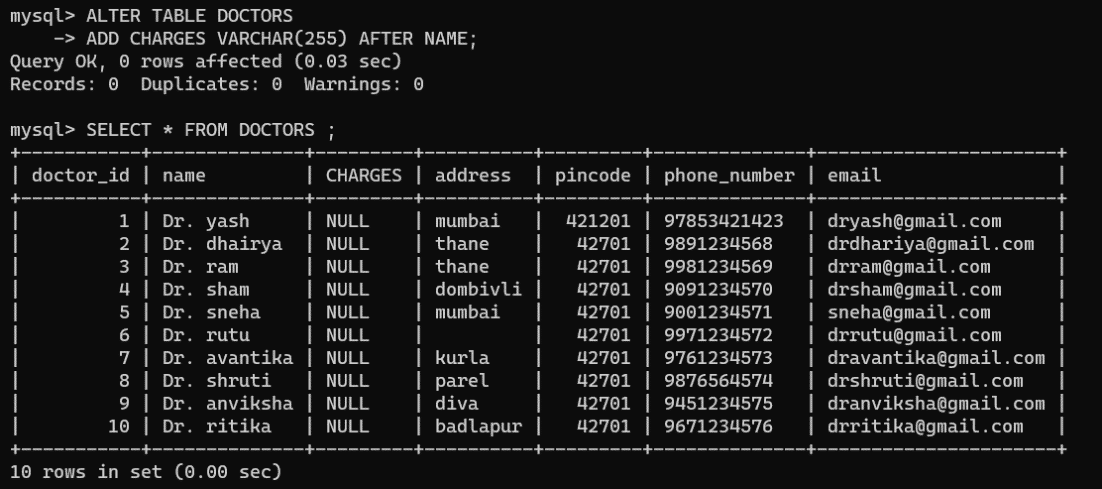
(14, 'POL901234', 'D Insurance','2024-02-01', '2025-02-01'),

(15, 'POL567890', 'C Insurance','2024-03-01', '2025-03-01');

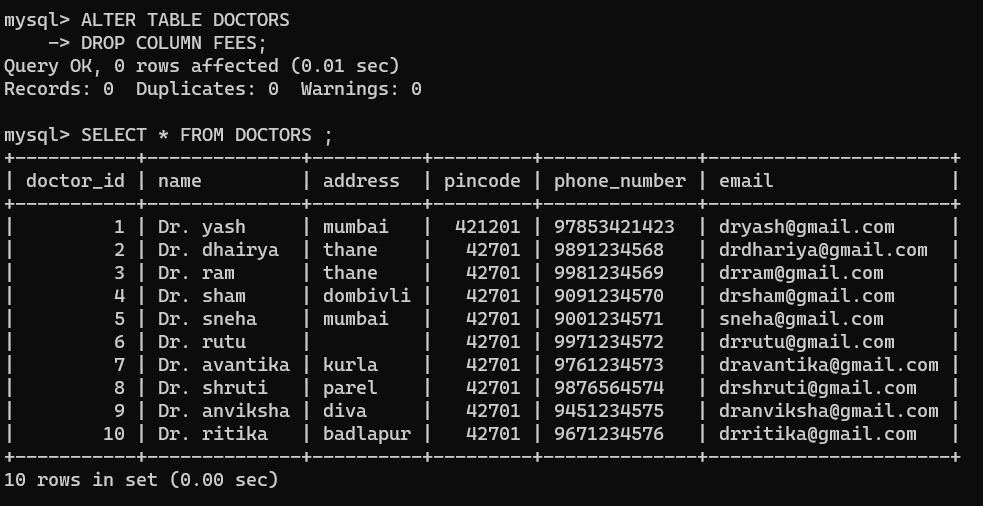


DDL (DATA definition language):

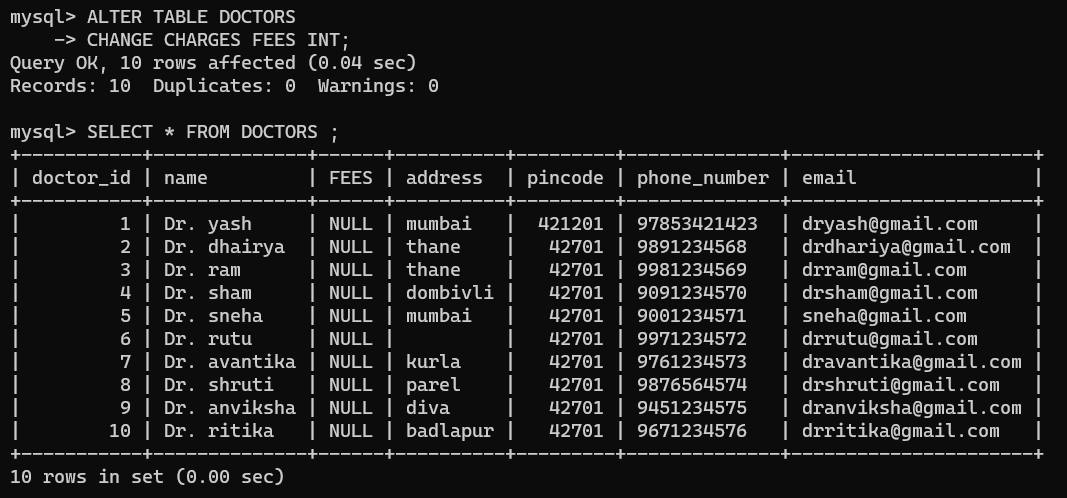
ADD NEW COLUMN IN DOCTORS TABLE (CHARGES):



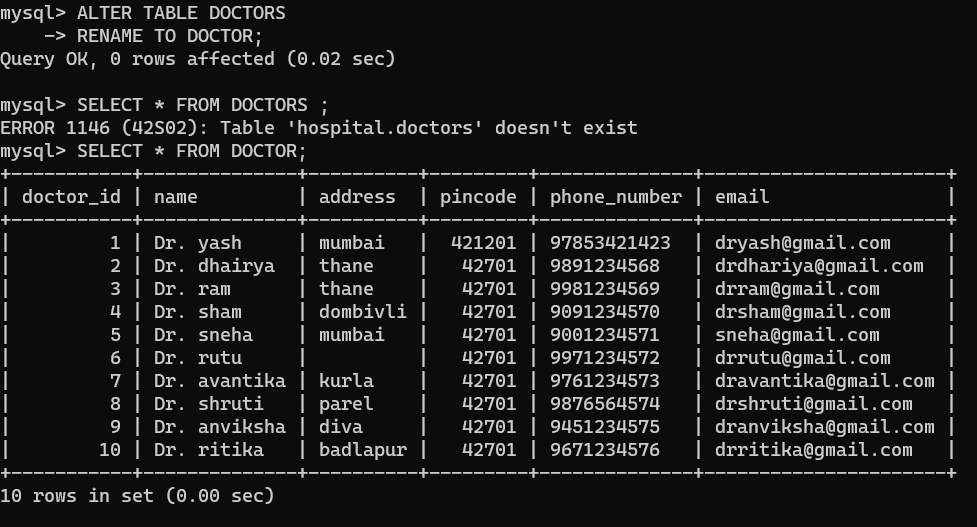
DELETE FEES COLUMN IN DOCTORS TABLE:



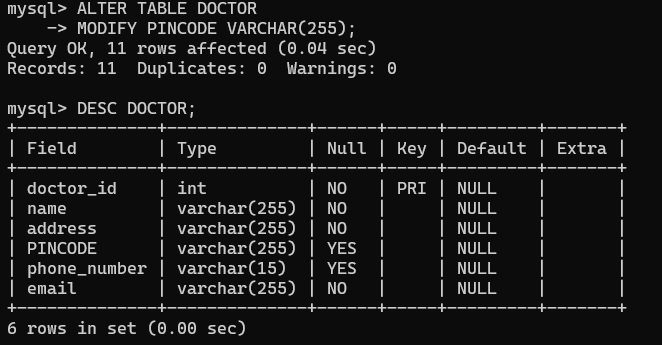
CHANGE COLUMN NAME FROM CHARGES TO FEES IN DOCTORS TABLE:



CHANGE TABLE NAME IN DOCTORS TABLE FROM DOCTORS TPO DOCTOR :



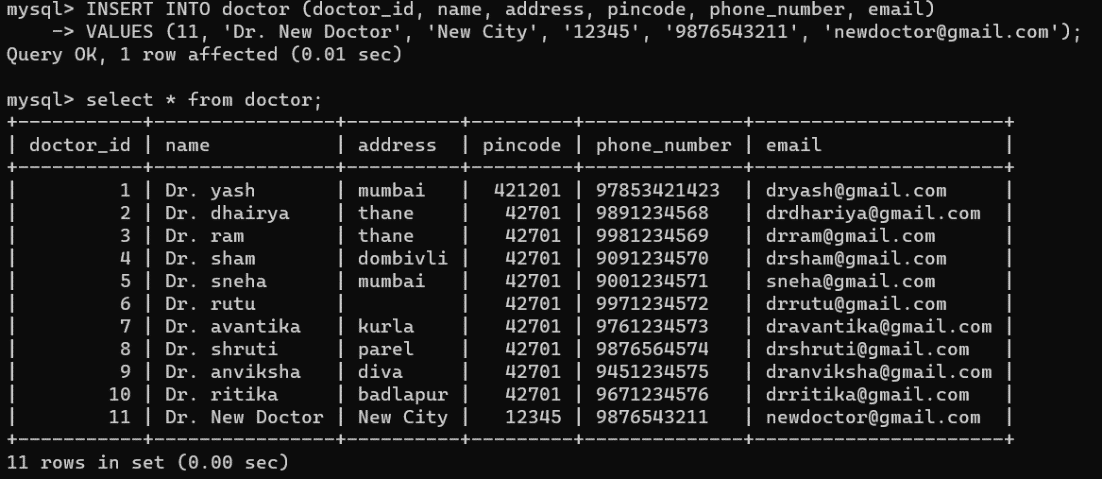
CHANGE DATATYPE OF COLUMN PINCODE FROM INT TO VARCHAR (255):



DML (DATA MANIPULATION LANGUAGE):

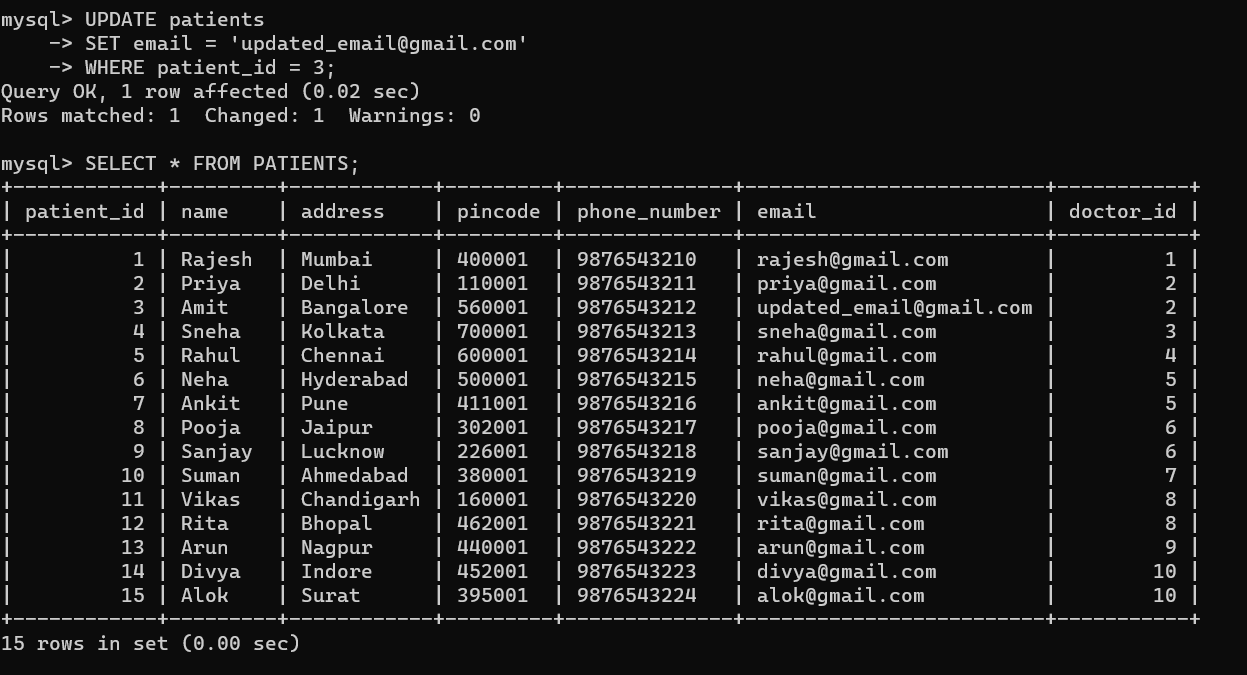
INSERT:

ADD A NEW DATA DR. NEW DOCTOR FROM NEW CITY PINCODE 12345, PHONE NO. 9876543211 AND [newdoctor@gmail.com](mailto:newdoctor@gmail.com) in doctor table :



UPDATE:

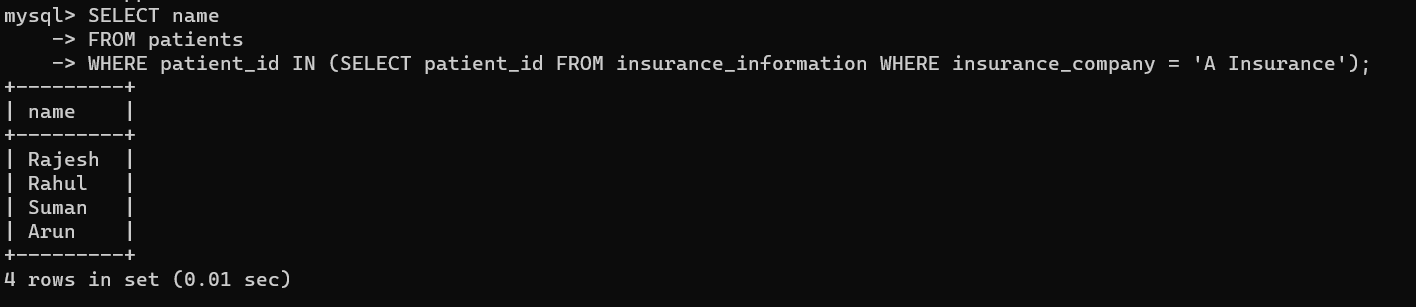
CHANGE EMAIL update\_email@gmail.com OF PATIENT\_ID =3 in PATIENT TABLE



DQL (DATA QUERY LANGUAGE):

1)SELECT \*:

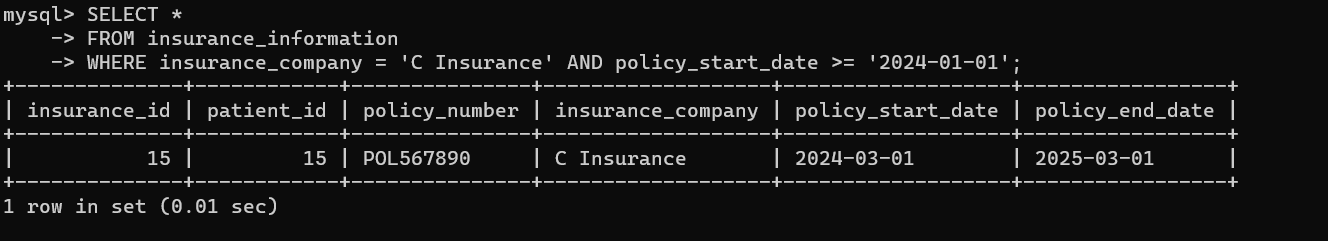
Retrieve the names of patients who have insurance policies from 'A Insurance' in the "insurance\_information" table:



CLAUSES:

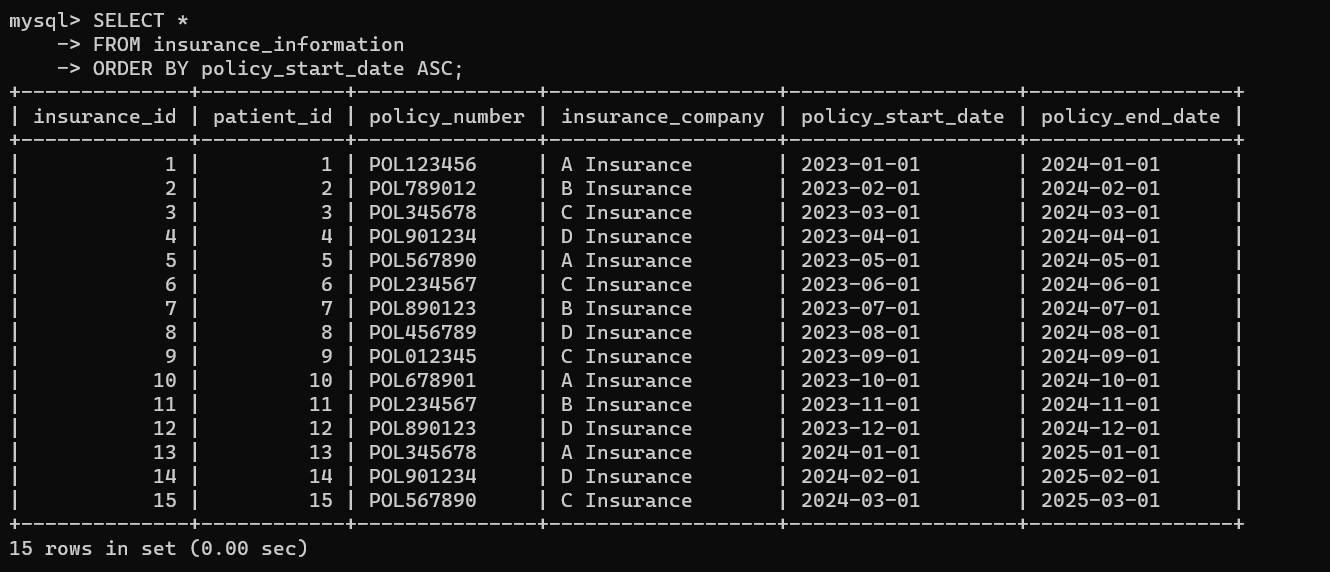
1)WHERE CLAUSE:

Retrieve insurance policies that belong to patients in 'C Insurance' and have a policy start date in or after '2024-01-01':



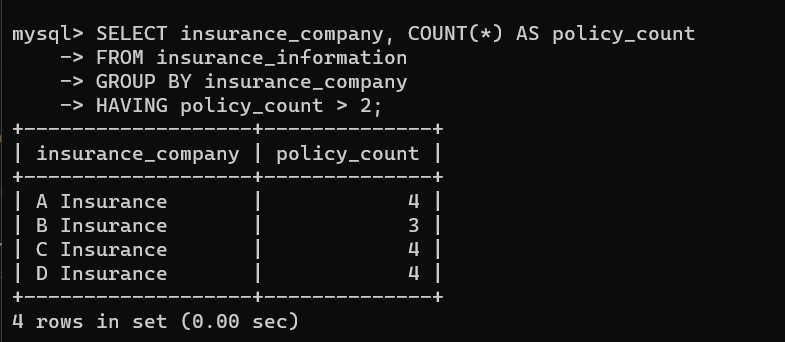
2)OERDER BY

Retrieve insurance policies ordered by policy start date in ascending order:



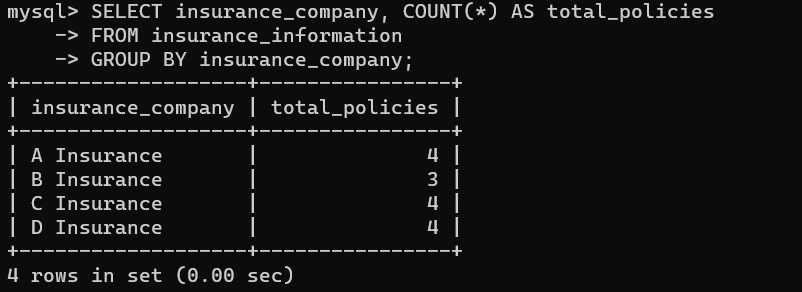
3)HAVING CLAUSE:

Find insurance companies with more than two policies and their count:



4)GROUP BY:

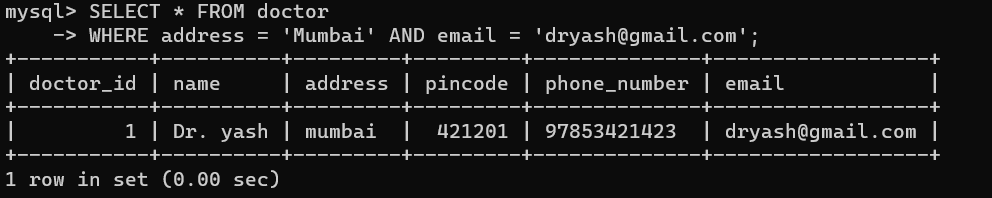
Calculate the total number of policies per insurance company:



OPERATOR FUNCTIONS:

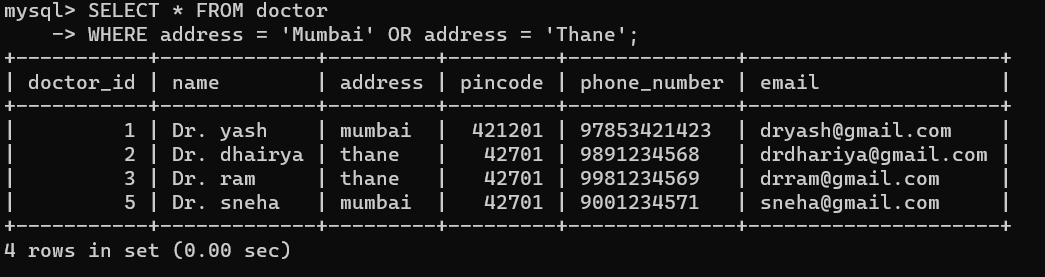
1) **AND Operator**

Retrieve doctors in Mumbai with a specific email:



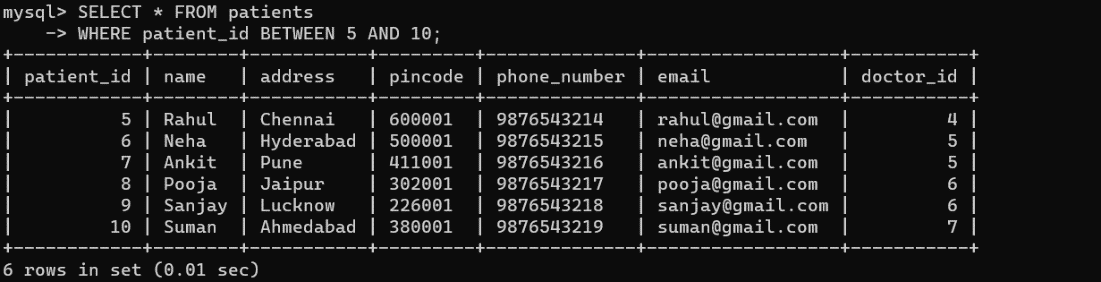
2) **OR Operator**

Retrieve doctors in Mumbai or Thane:



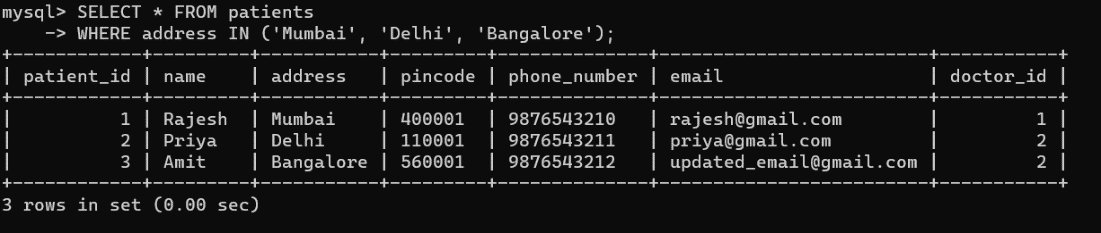
3) **BETWEEN Operator**

Retrieve patients with IDs between 5 and 10:



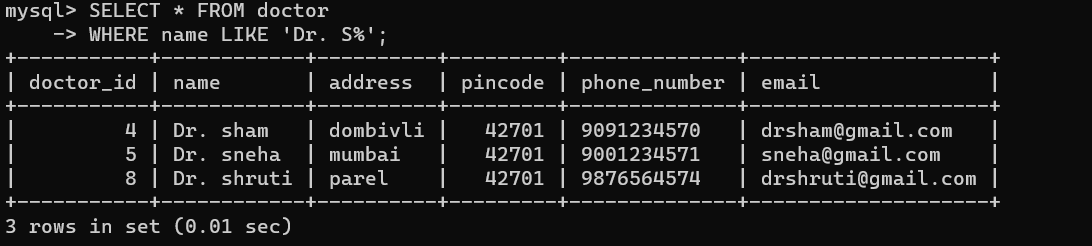
4) **IN Operator**

Retrieve patients from specific cities:



5) **LIKE Operator**

Retrieve doctors with names starting with "Dr. S":



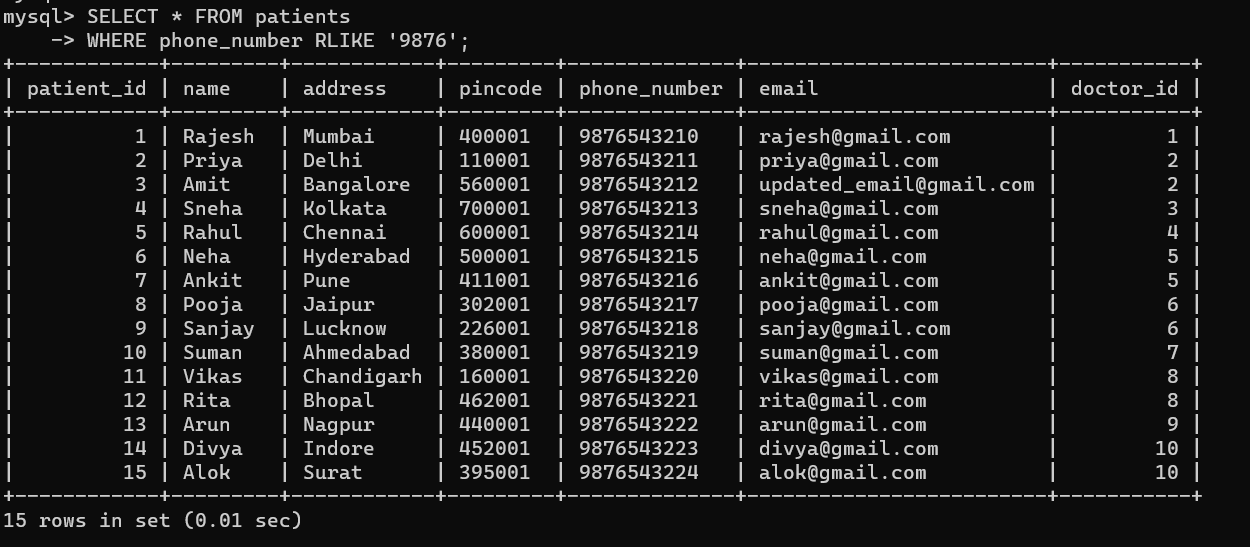
6) **DISTINCT OPERATOR**

Retrieve unique addresses of patients:



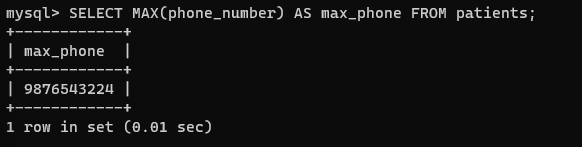
7) **RLIKE Operator**

Retrieve patients with phone numbers containing "9876":

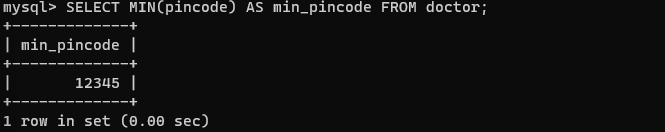


AGGREGATE FUNCTION

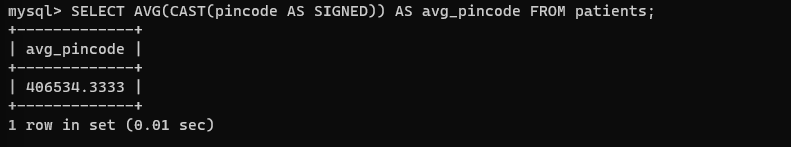
1) **MAX** - Find the maximum phone number among patients:



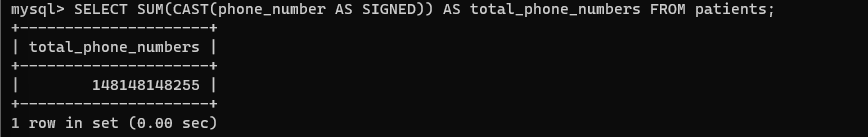
2) **MIN** - Find the minimum pincode among doctors:



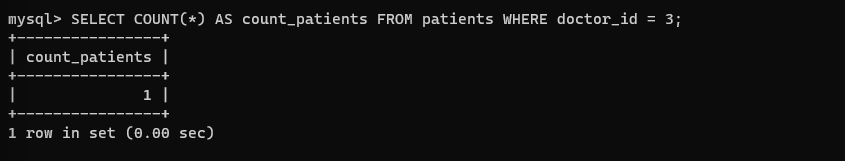
3) **AVG** - Calculate the average pincode among patients:



4) **SUM** - Calculate the total phone numbers of patients:



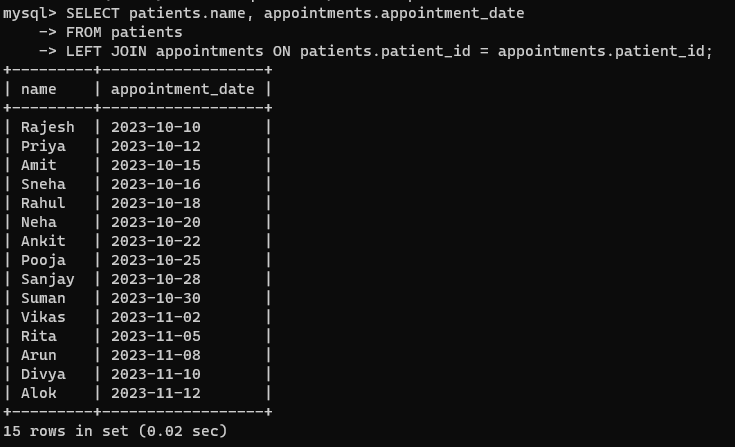
5) **COUNT** - Count the number of patients with a doctor ID of 3:



JOINS:

**LEFT JOIN**

List the names of all patients and their appointment dates, even if they haven't scheduled any appointments. Which patients have no appointments



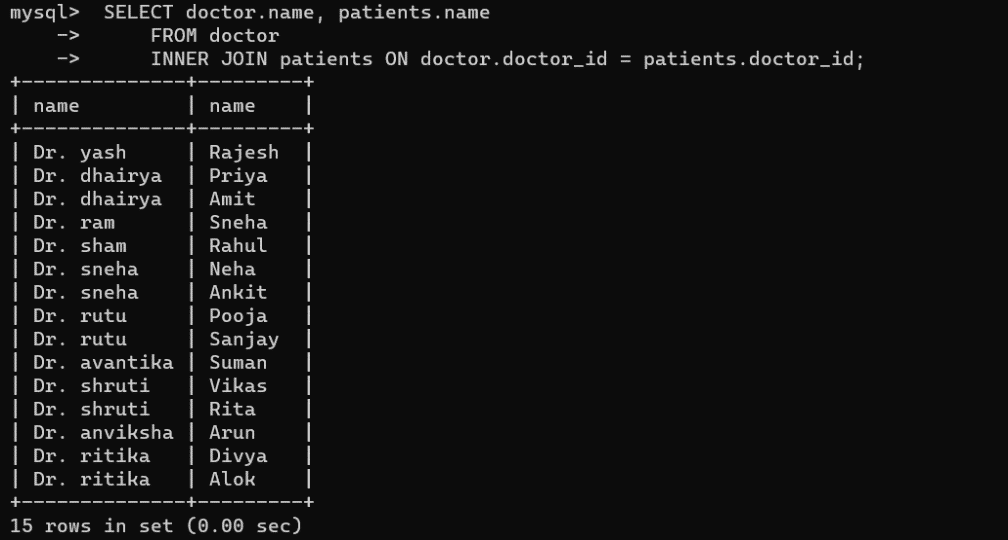
**RIGHT JOIN**

Retrieve the names of all patients who have scheduled appointments and their corresponding appointment dates. Are there any appointments without associated patients



**INNER JOIN**

Find the names of doctors and their corresponding patients. Which doctors are currently treating patients, and who are these patients



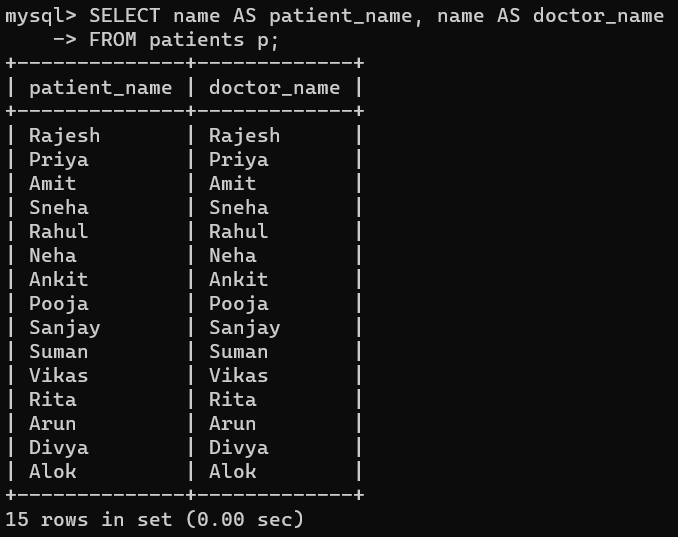
**CROSS JOINS**

Combine the names of all patients with the names of all doctors to explore all possible patient-doctor pairs. How many unique combinations can be formed



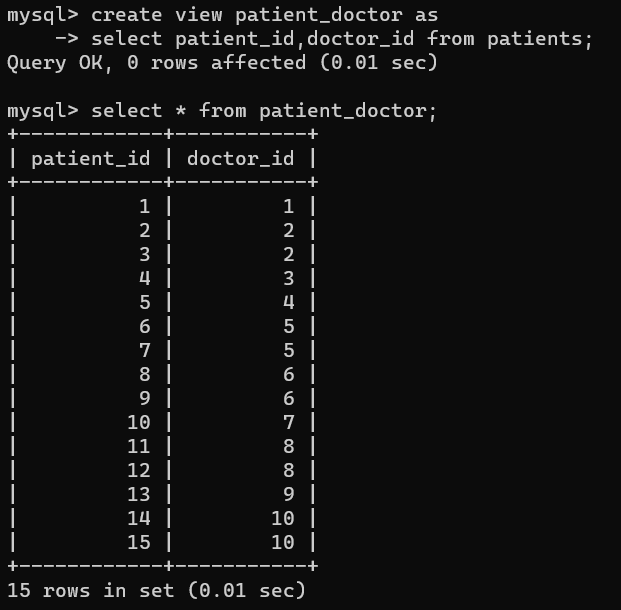
**ALLIAS:**

Print the data using Allias of patient table



View:

Write a query to create a temporary table to show the which patient is treated by which doctor



**Window Functions**

### 1. Ranking the Doctors based on the number of appointments:

SELECT

doctor\_id,

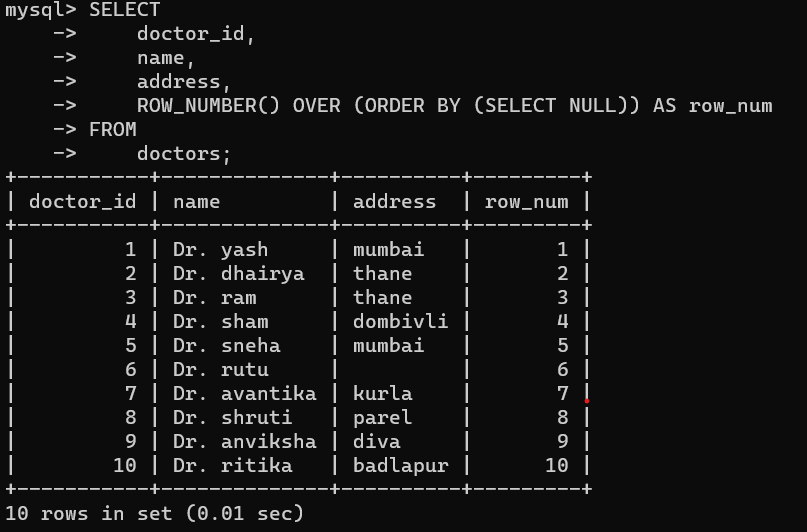
name,

address,

ROW\_NUMBER() OVER (ORDER BY (SELECT NULL)) AS rank

FROM

doctors;



### 2. Calculating the Total Number of Patients for Each Doctor:

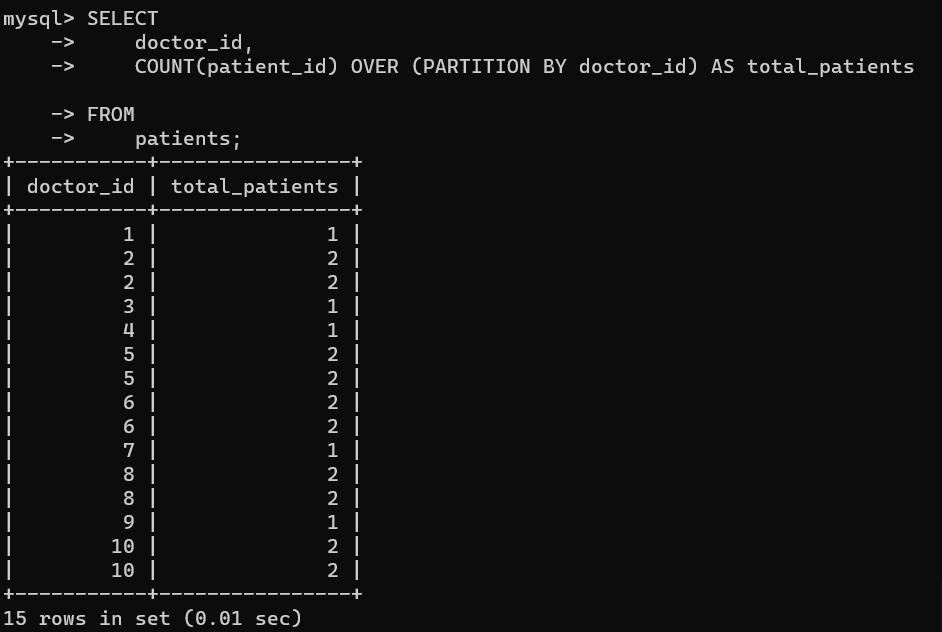
SELECT

doctor\_id,

COUNT(patient\_id) OVER (PARTITION BY doctor\_id) AS total\_patients

FROM

patients;



### 3. Finding the Cumulative Sum of Policy End Dates for Each Insurance Company:

SELECT

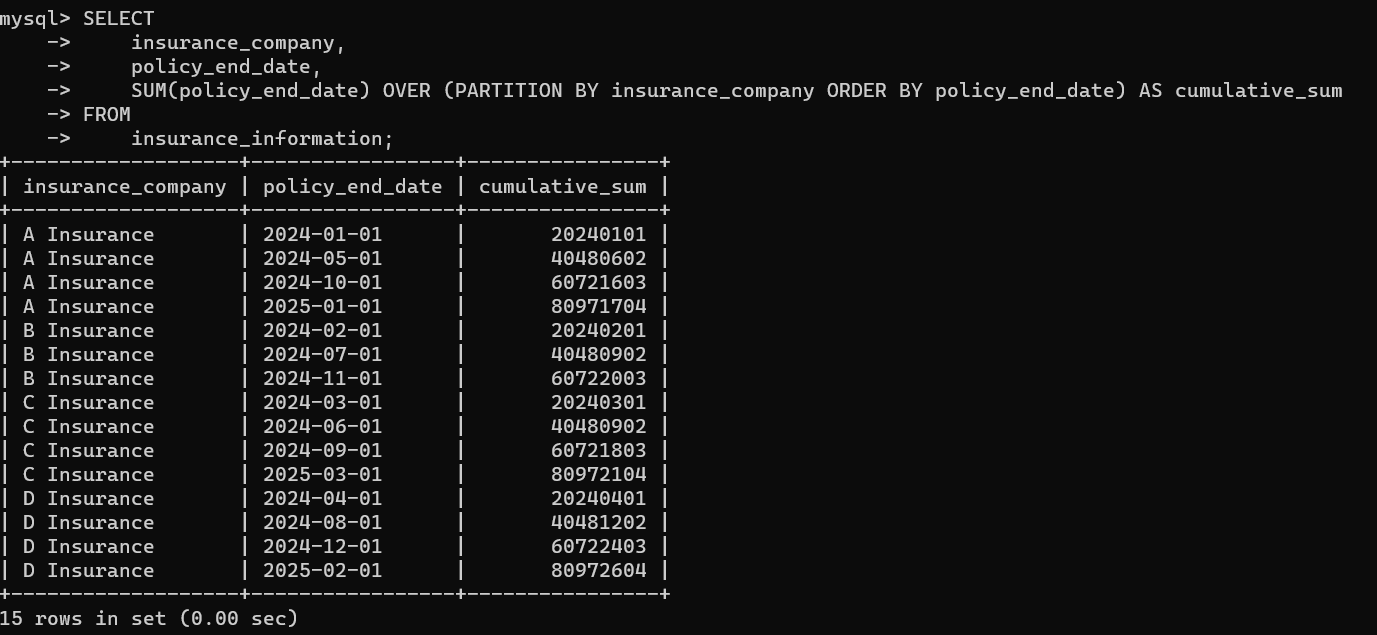
insurance\_company,

policy\_end\_date,

SUM(policy\_end\_date) OVER (PARTITION BY insurance\_company ORDER BY policy\_end\_date) AS cumulative\_sum

FROM

insurance\_information;



### 4. Finding the Maximum Appointment Date for Each Doctor:

SELECT

doctor\_id,

MAX(appointment\_date) OVER (PARTITION BY doctor\_id) AS max\_appointment\_date

FROM

appointments;

