Hospital Management Application

Srinath Penugonda

Github: <https://github.com/Srinath2905/Hospital-Management-System.git>

05-1-2022

**Initial Proposal**

During the course I would like to develop a hospital management application. In today’s world corporate hospitals are more in demand, so the application needs to be maintained efficiently. Hospital management database contains data mainly related to Patients, Doctors, Appointments, Billing, Diagnose, Labs, Feedback.

The data stored here is very important. Most of the patients schedule an appointment with the doctor but, there are few patients who are unable to schedule an appointment due to unexpected health conditions and needs a doctor and attendant employees. In such scenario we can make use of database and retrieve which doctor and staff can attend the patient immediately. And also using feedback details from the database, increments and promotions can be allocated to hospital staff.

Few patients will pay the bill directly but few pay through installments and few claim insurance. So, we will be using the live data and also data flowing in from banks, insurance groups, laboratories. All the data should be collected and shown under a single application.

Mostly the data is used by the head group of the organization in the form of reports to analyze whether the hospital is running on the profit end or not. And also, to check where it stands in the competition of corporate hospitals.

**Relational Database Design Process**

As already mentioned in the Initial proposal the below are my main entities and their attributes.

Entity 1

|  |  |
| --- | --- |
| **PATIENT** | DATA TYPE |
| PatientID | INT |
| PatientRegdNo | VARCHAR(45) |
| FirstName | VARCHAR(45) |
| LastName | VARCHAR(45) |
| DateOfBirth | DATE |
| Gender | VARCHAR(10) |
| PhoneNumber | VARCHAR(12) |
| EmailID | VARCHAR(45) |
| BloodGroup | VARCHAR(10) |

Entity 2

|  |  |
| --- | --- |
| **DOCTOR** | DATATYPE |
| DoctorID | INT |
| DoctorName | VARCHAR(45) |
| EmailID | VARCHAR(45) |
| Specialization | VARCHAR(45) |

Entity 3

|  |  |
| --- | --- |
| **APPOINTMENT** | DATATYPE |
| PatientID | INT |
| DoctorID | INT |
| AppoitmentDate | DATETIME |
| AppointmentType | VARCHAR(45) |

Entity 4

|  |  |
| --- | --- |
| **BILLING** | DATATYPE |
| PatientBillingID | INT |
| PatientRegisterID | VARCHAR(45) |
| TransactionDesc | VARCHAR(45) |
| Amount | DECIMAL(8,2) |
| GeneratedDate | DATETIME |
| PatientAddress | VARCHAR(100) |
| PaymentType | VARCHAR(45) |

Entity 5

|  |  |
| --- | --- |
| **LAB** | DATATYPE |
| PatientLabReportID | INT |
| PatientRegisterID | VARCHAR(45) |
| DiseaseName | VARCHAR(100) |
| LabTestID | INT |
| TestValue | VARCHAR(45) |
| Comment | VARCHAR(100) |

**Data Sources**

I have created three records per each entity using delimited comma separated values. Below mentioned are my five separate files each containing 3 records. I am attaching them in github.

Patient.csv

Doctor.csv

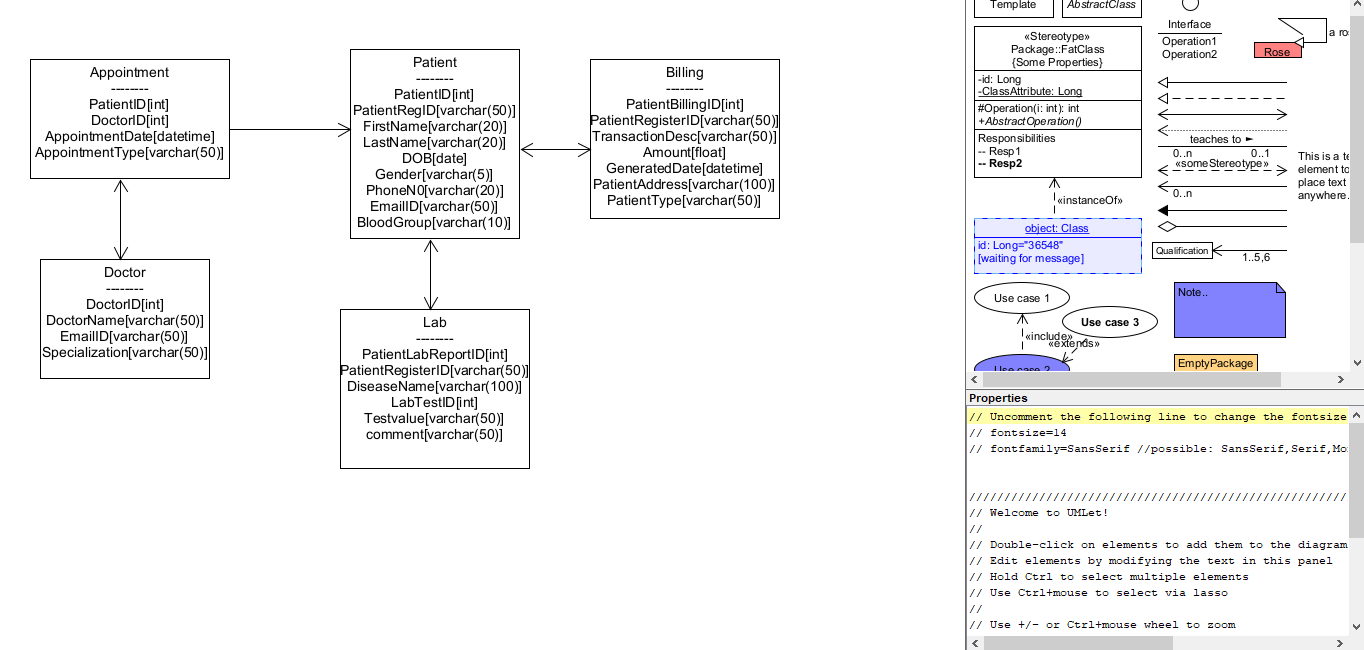
Appointment.csv

Billing.csv

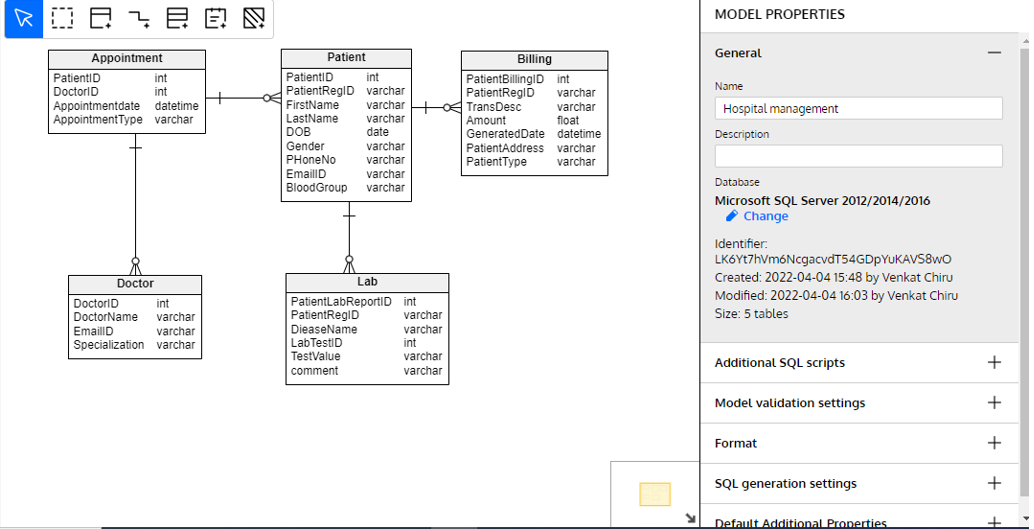
Lab.csv

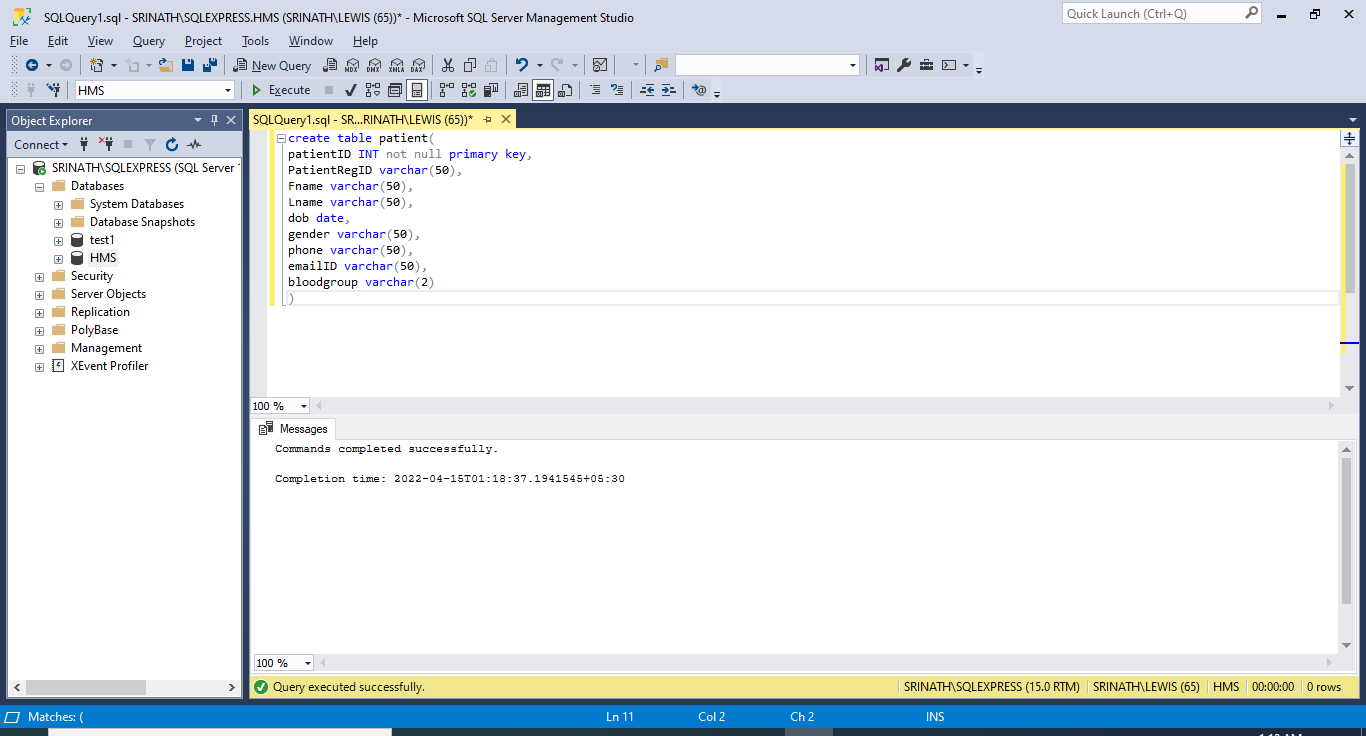
ER Diagram

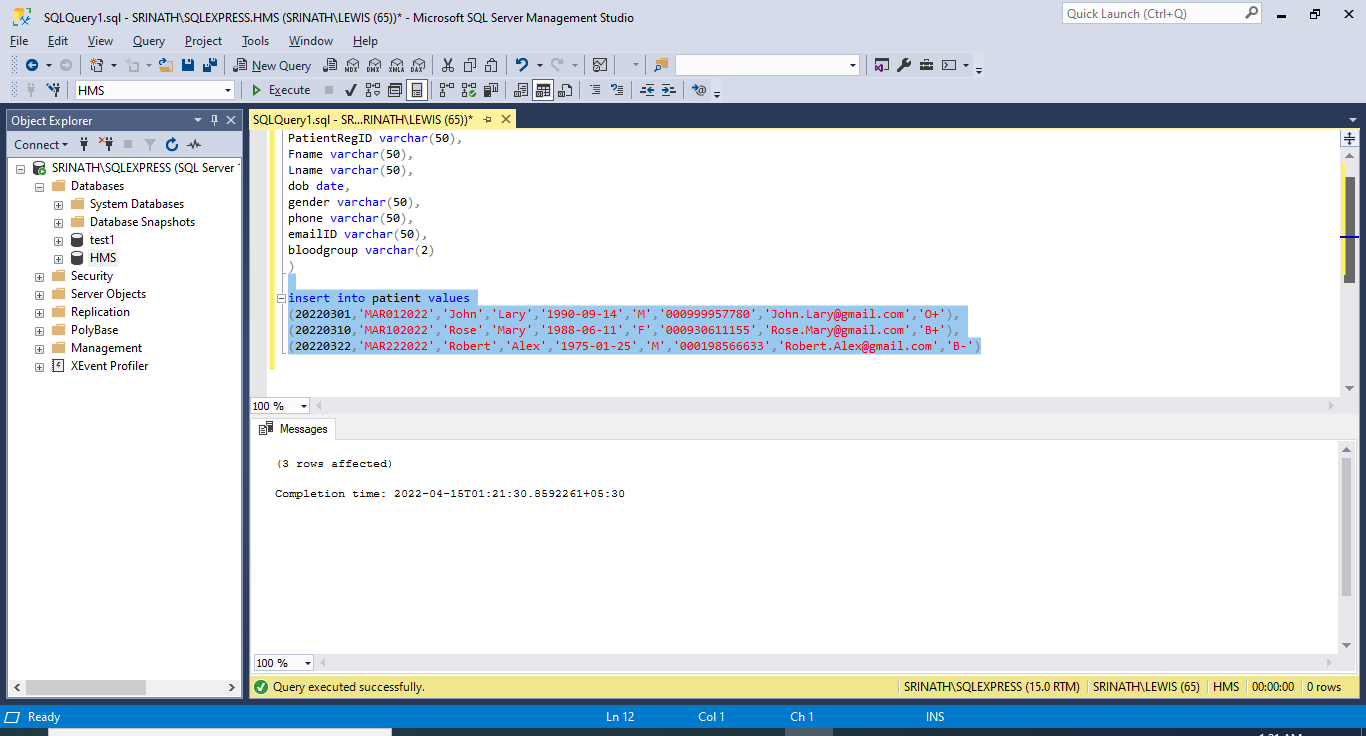
UMLet

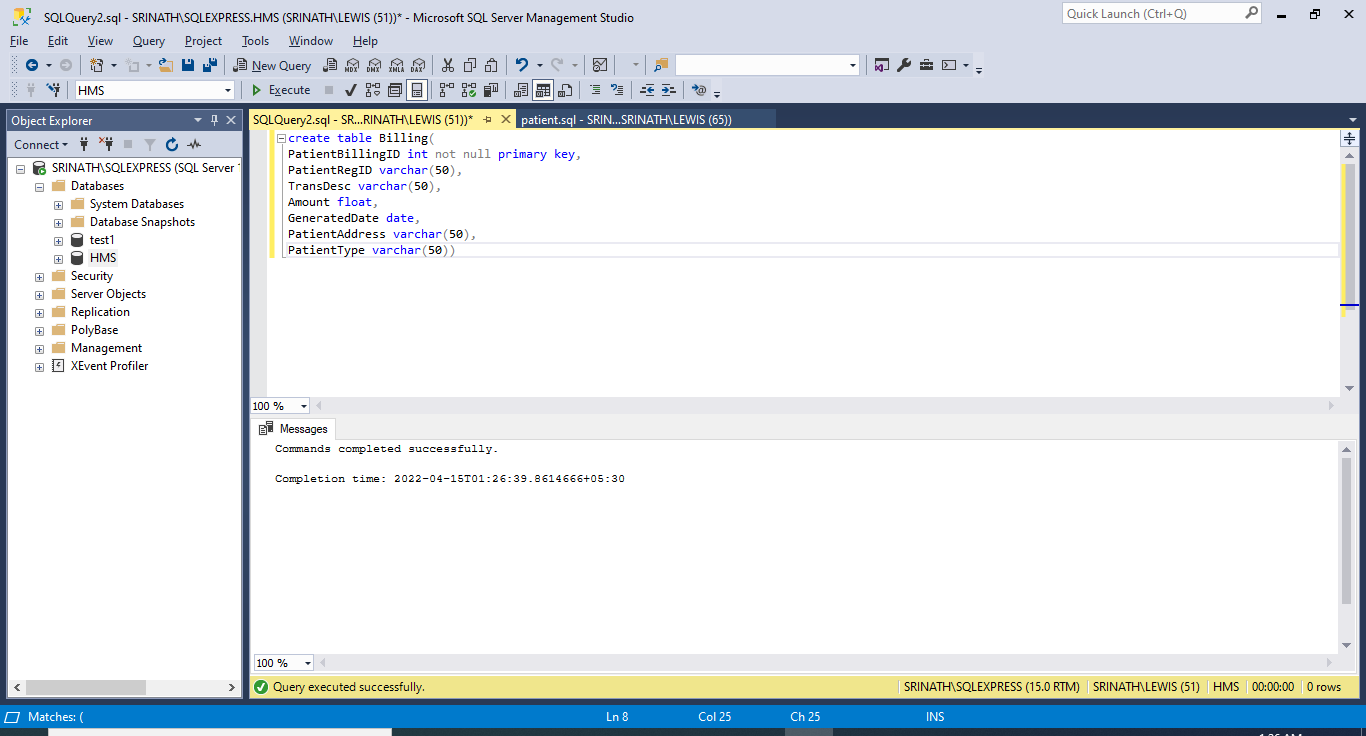


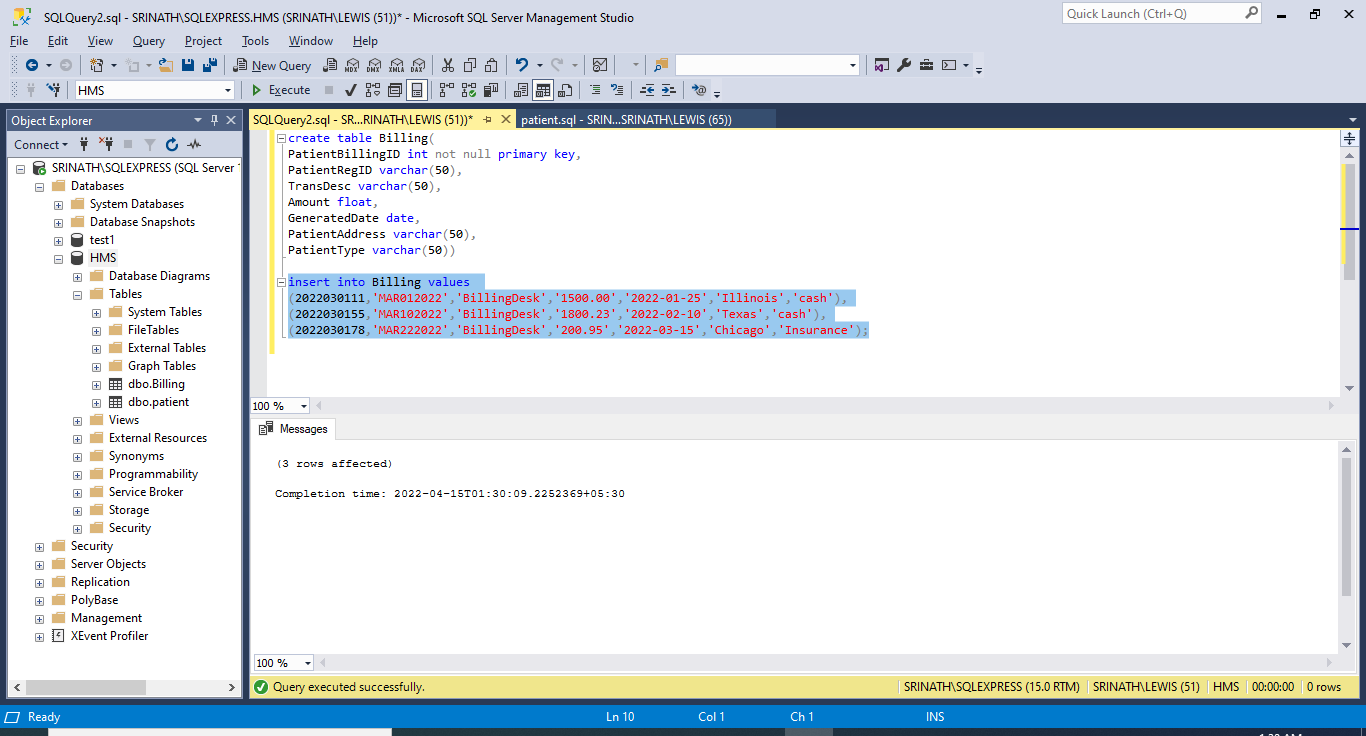
**Vertabelo**

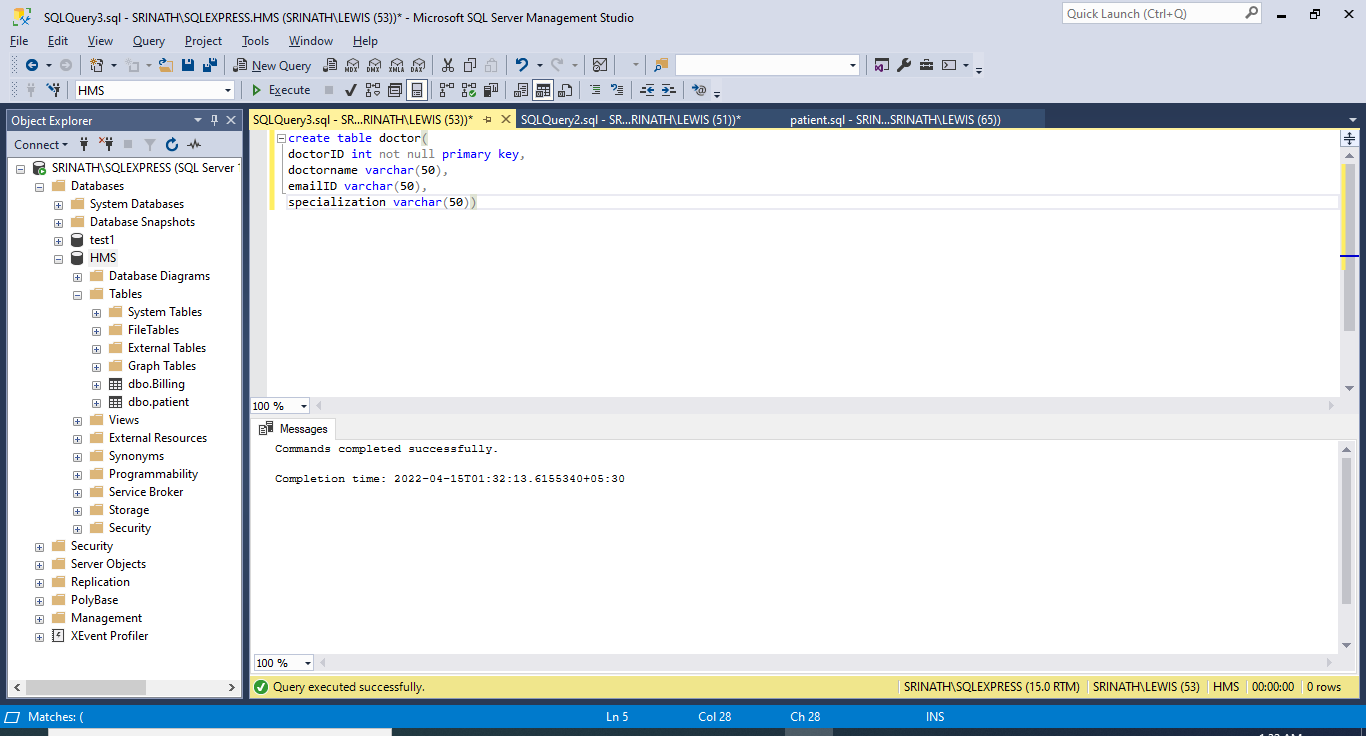


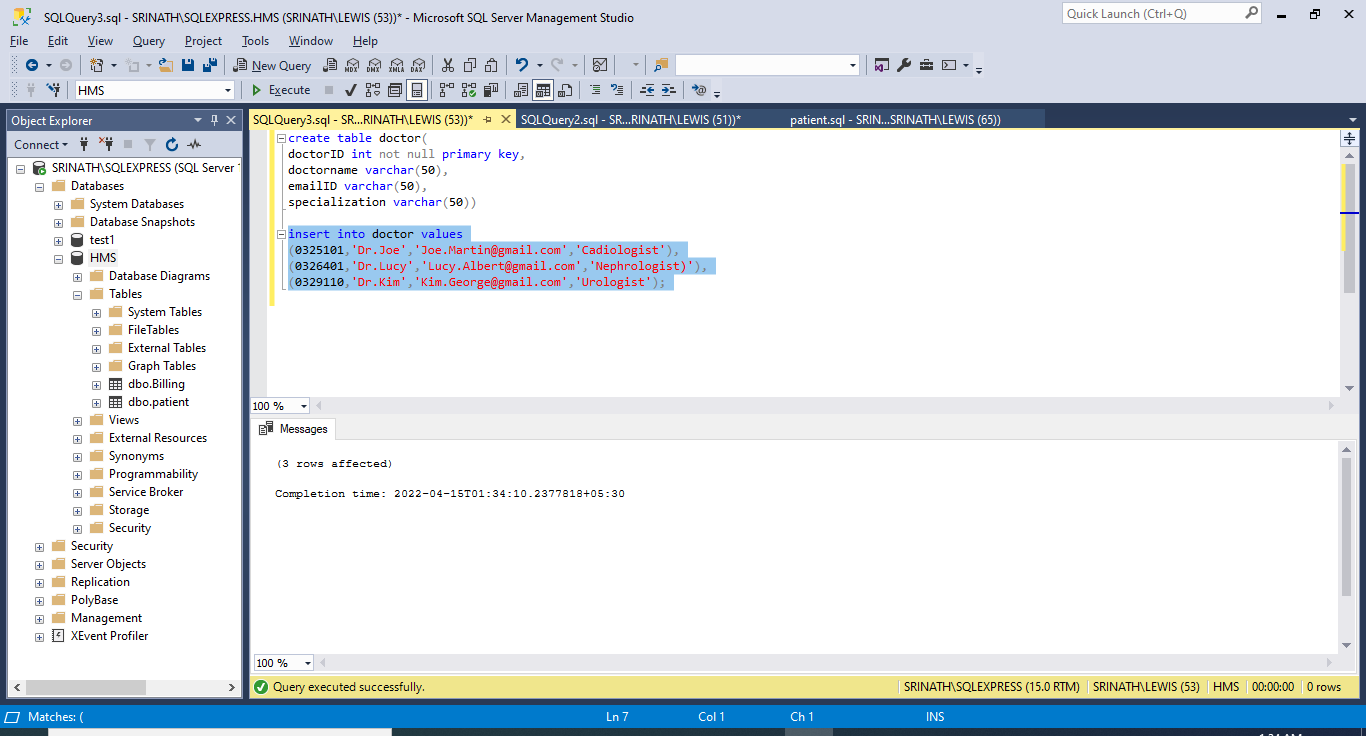
\*

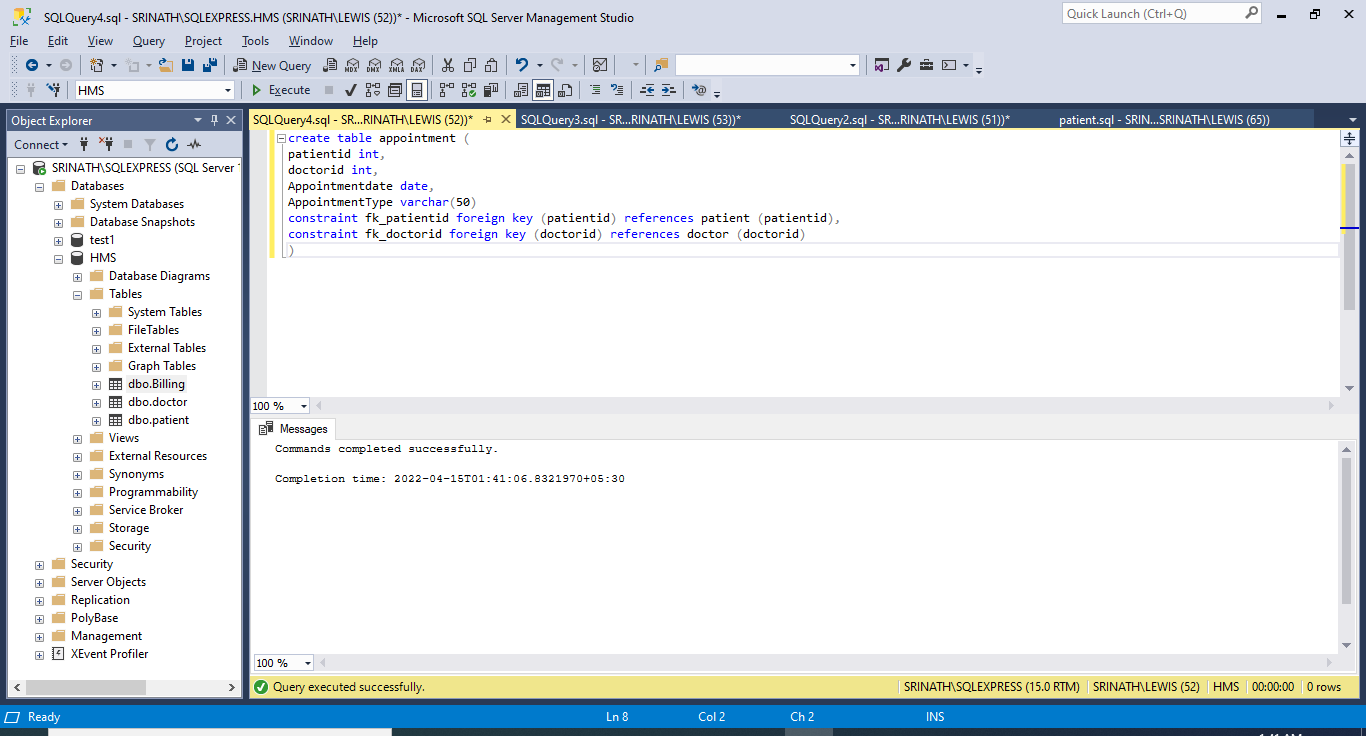


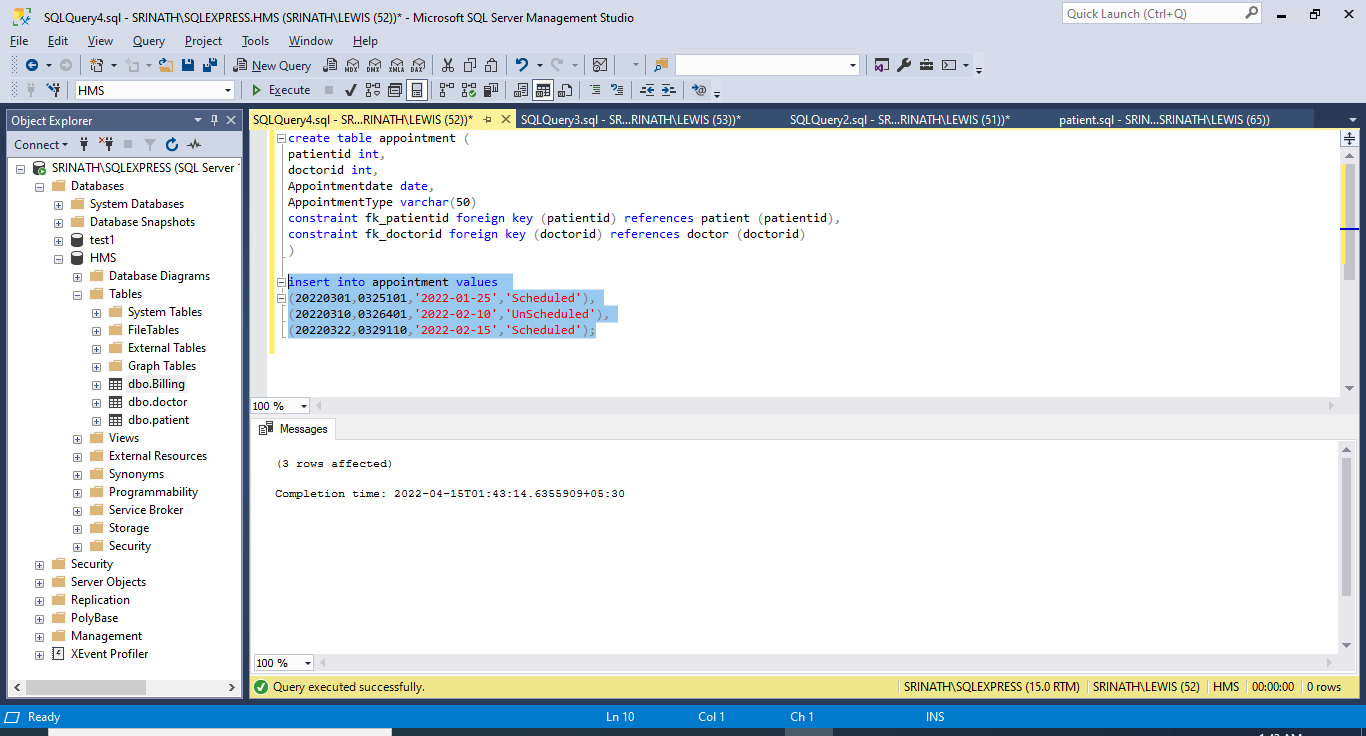


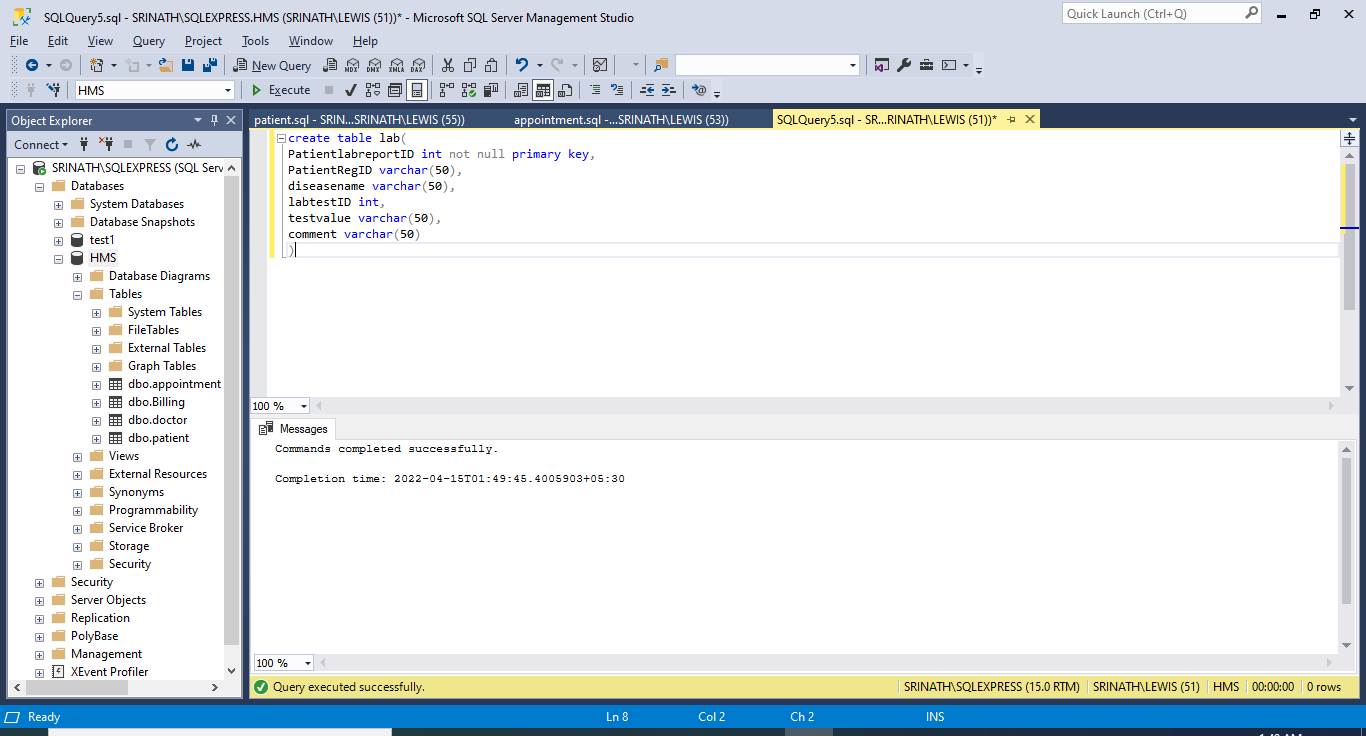


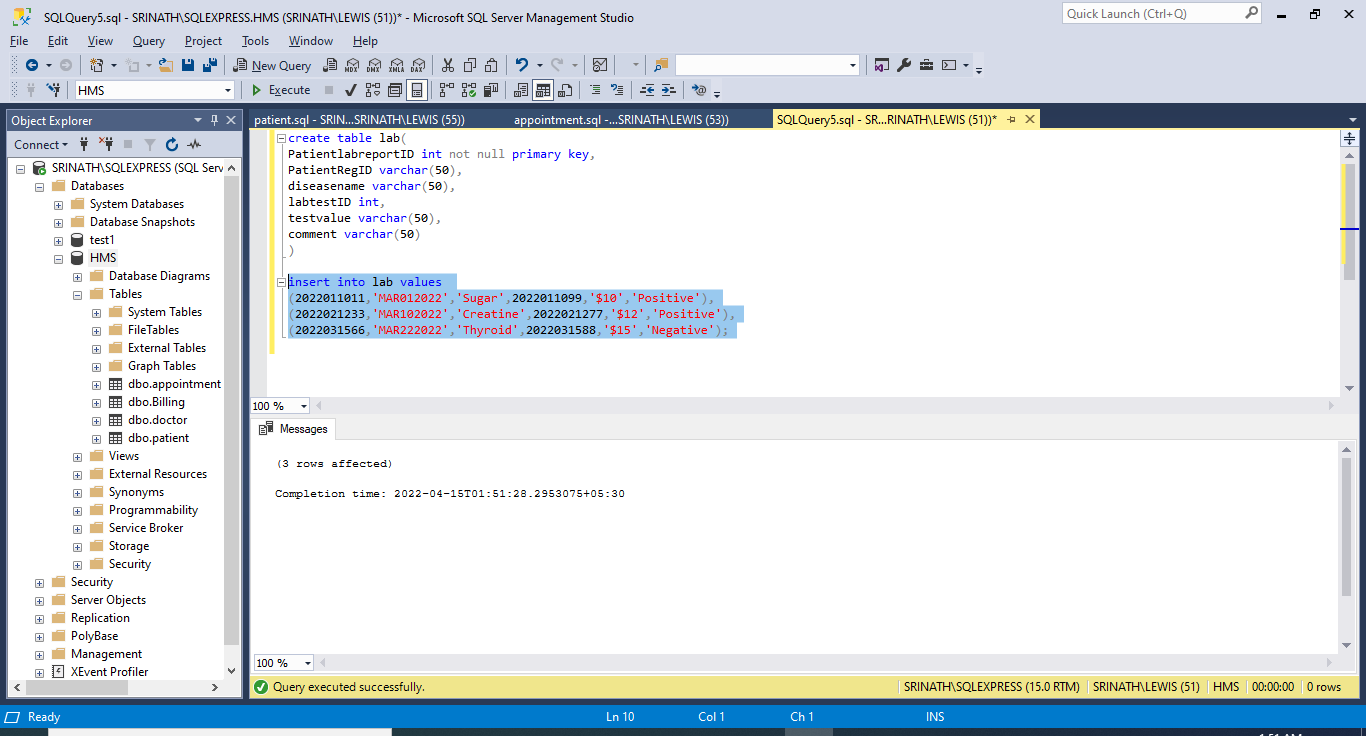


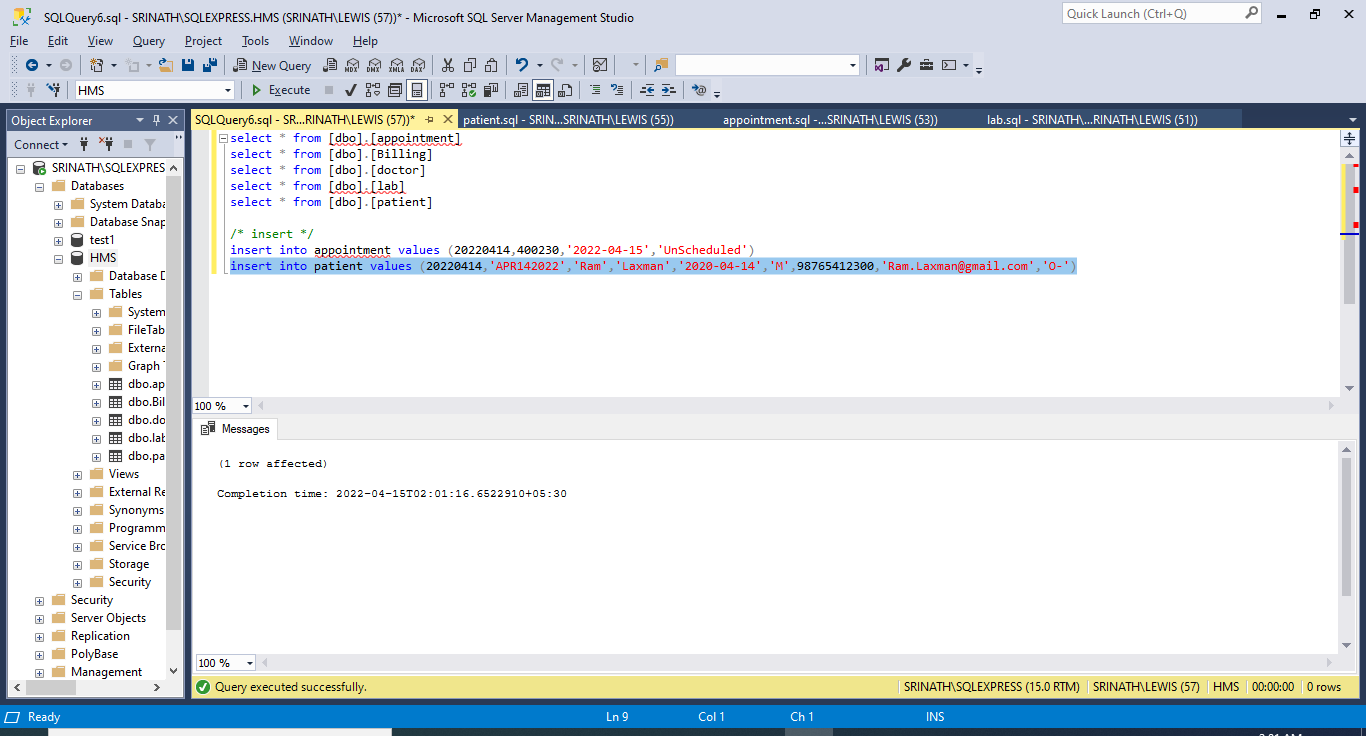


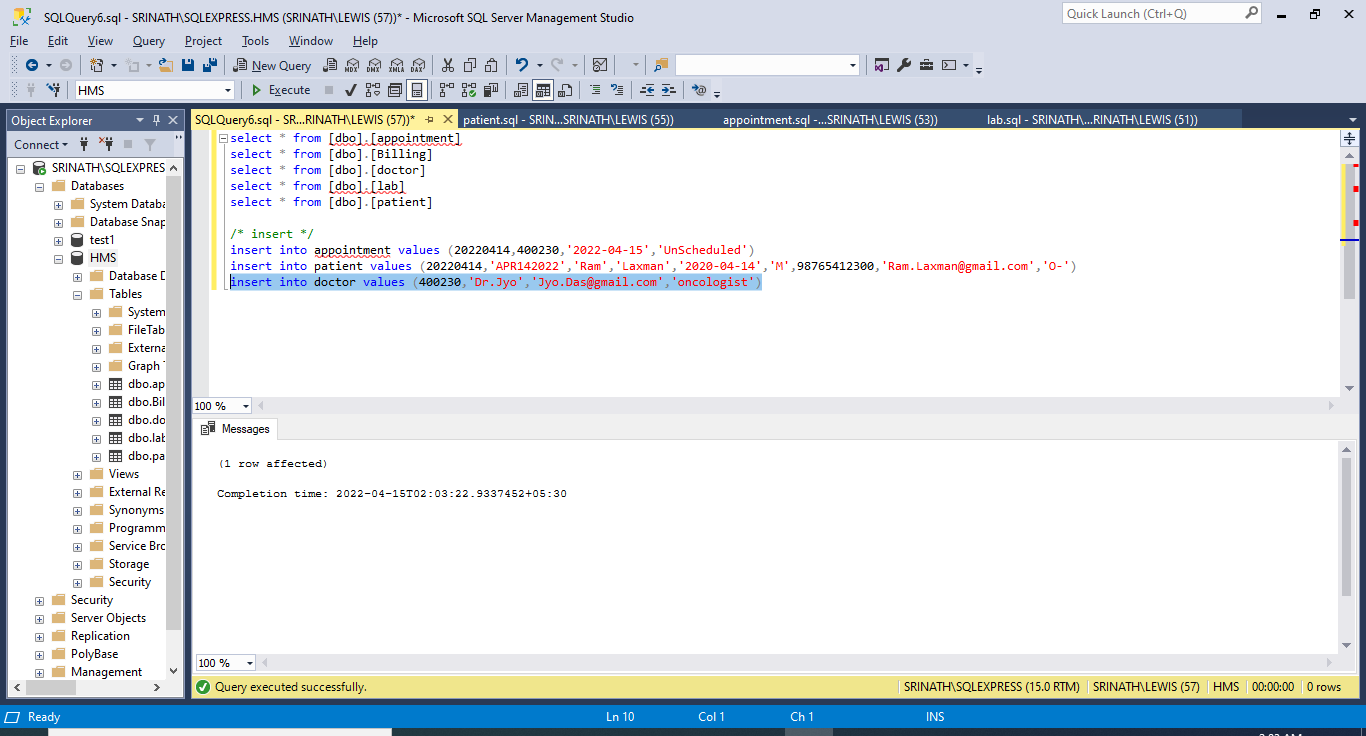


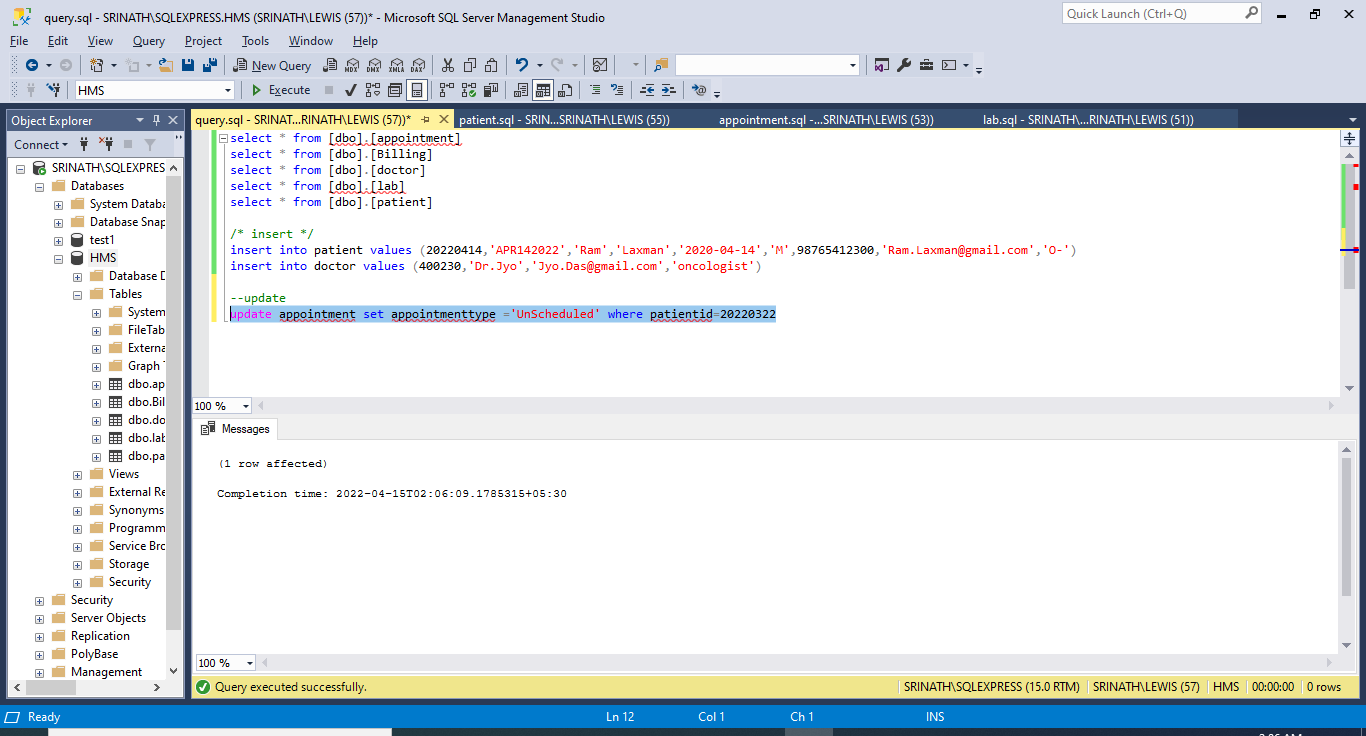


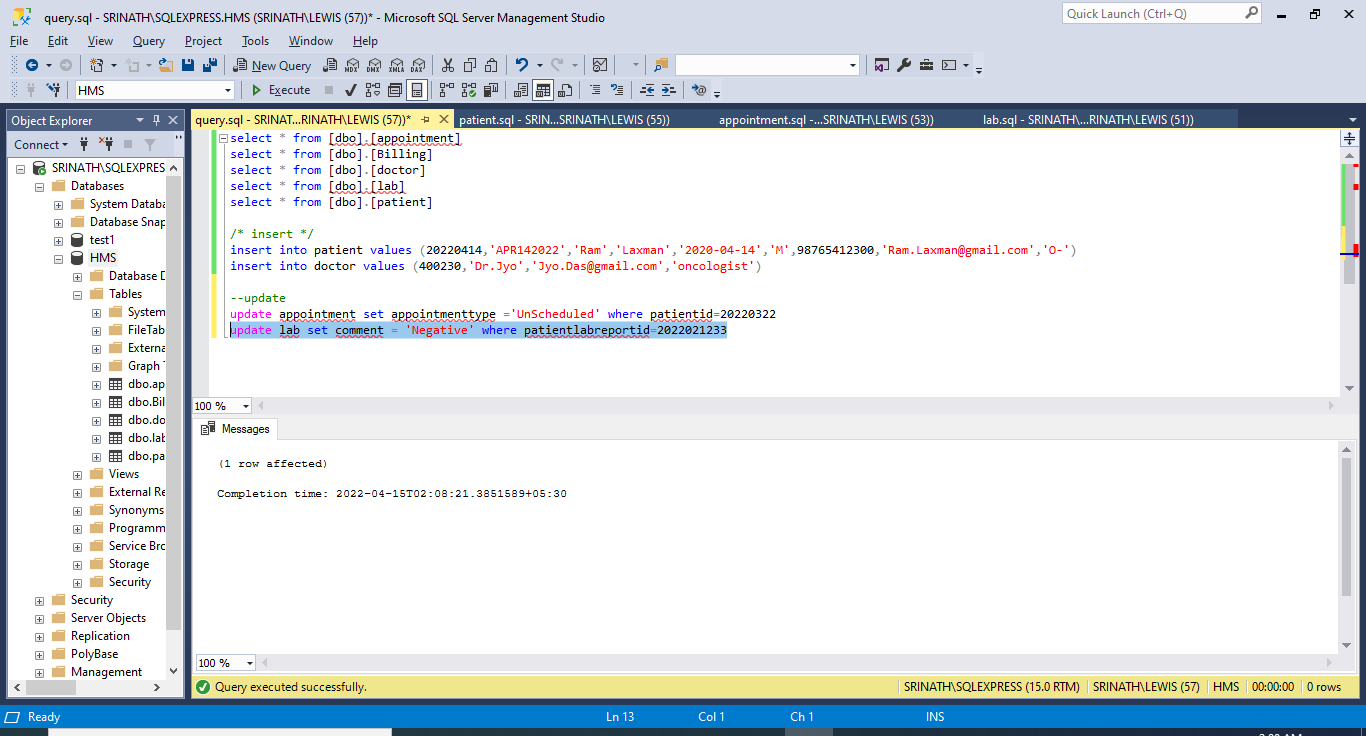


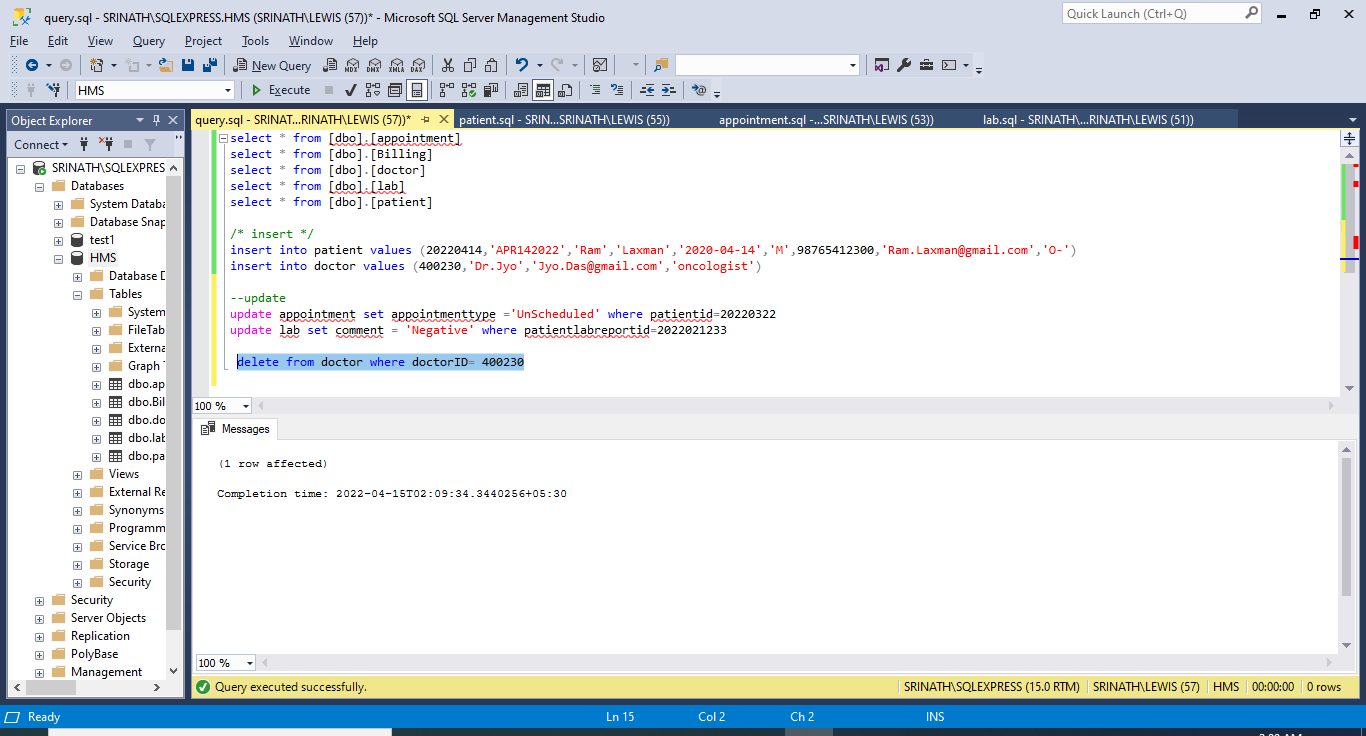


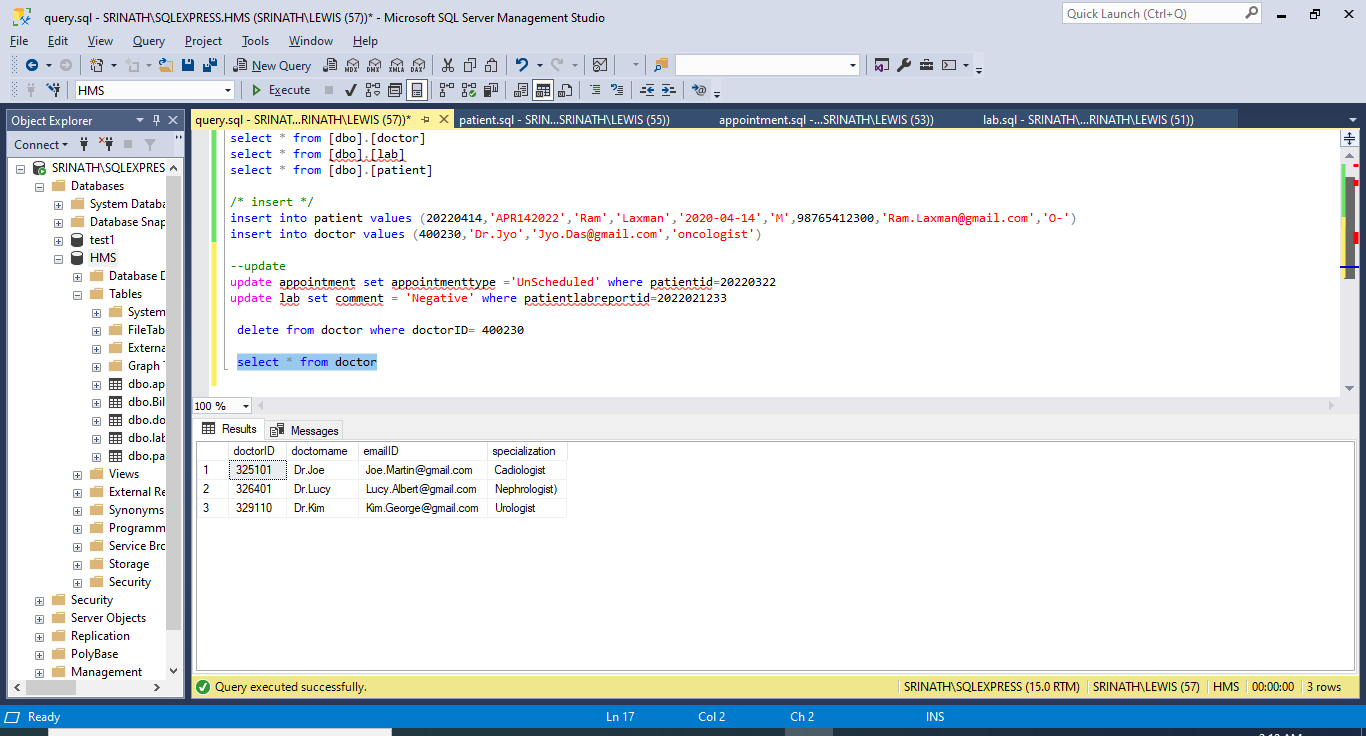


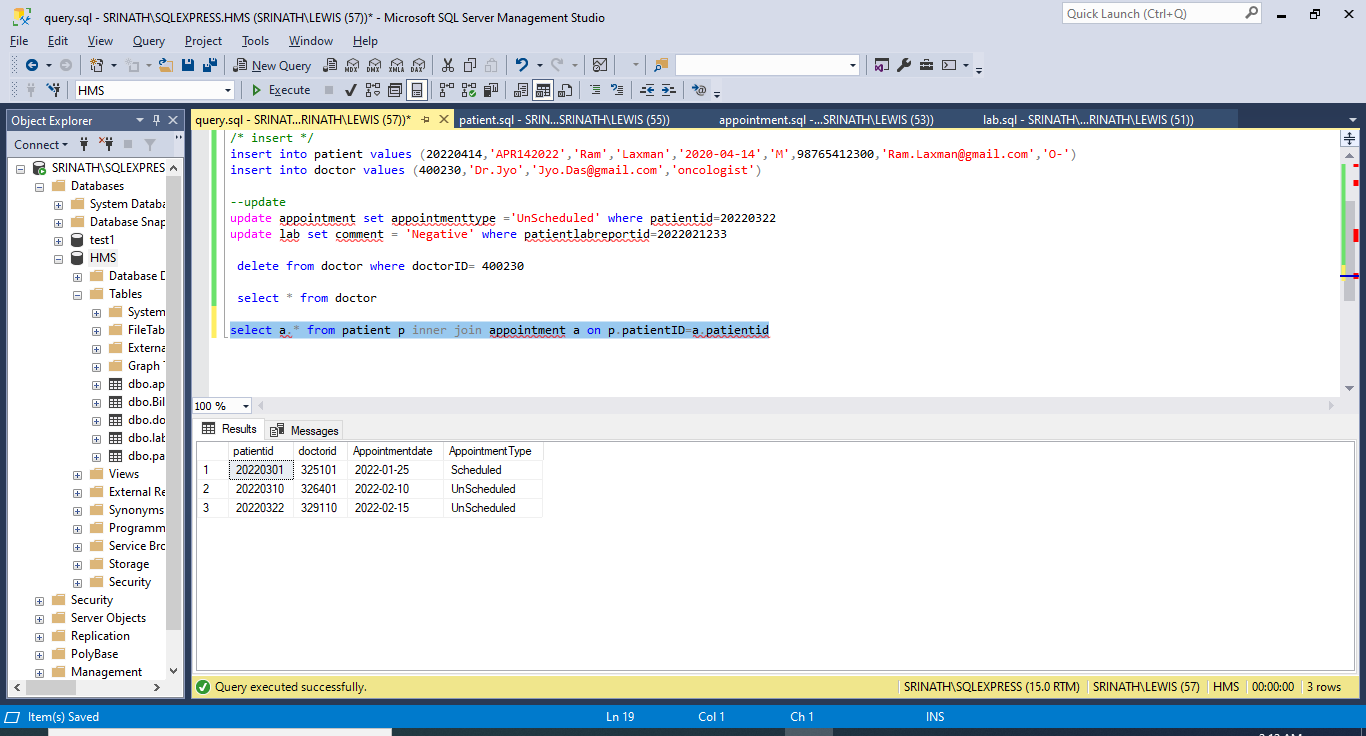


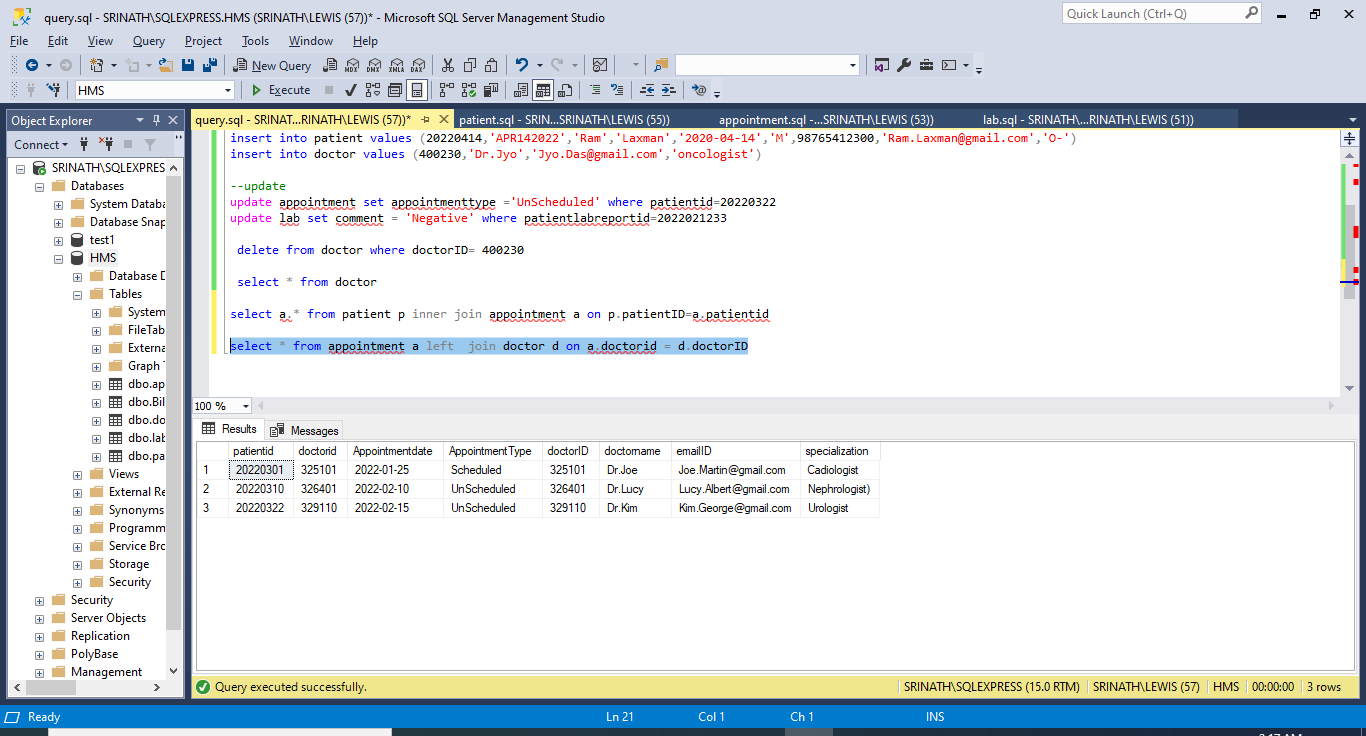


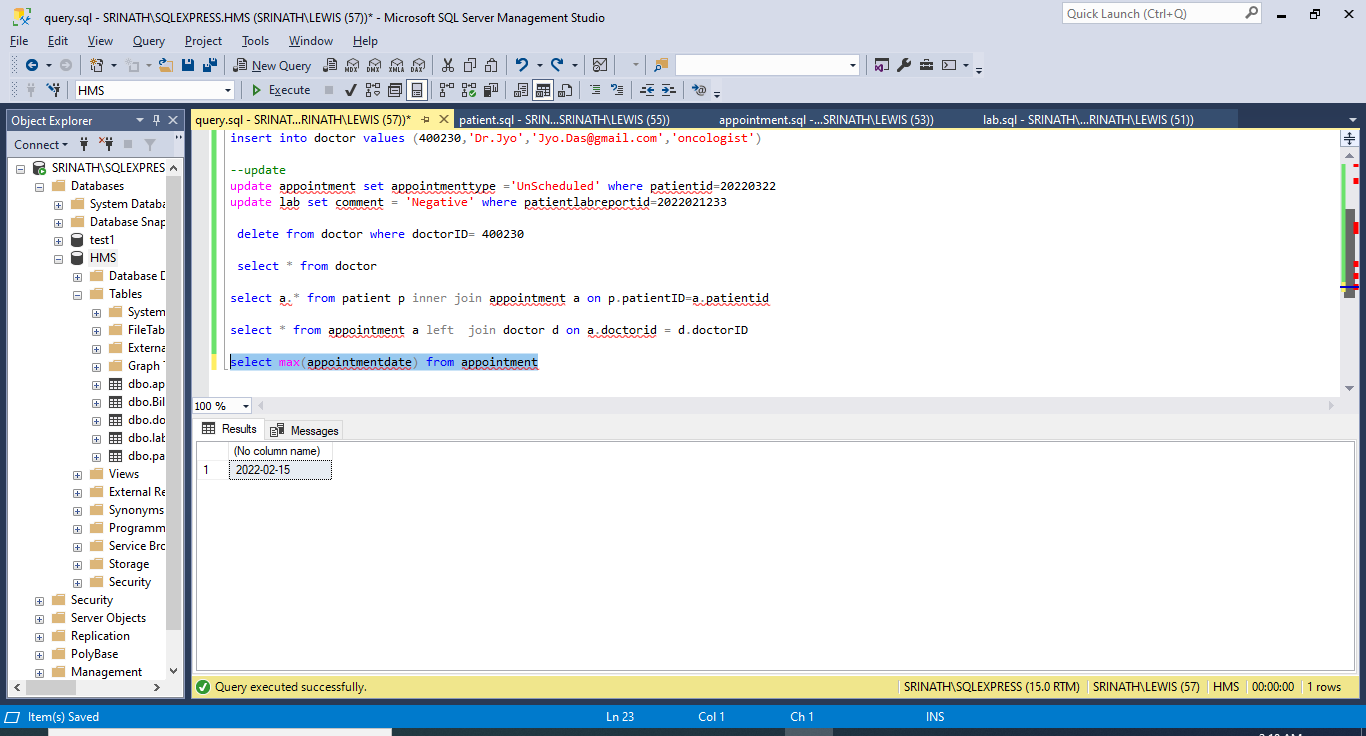


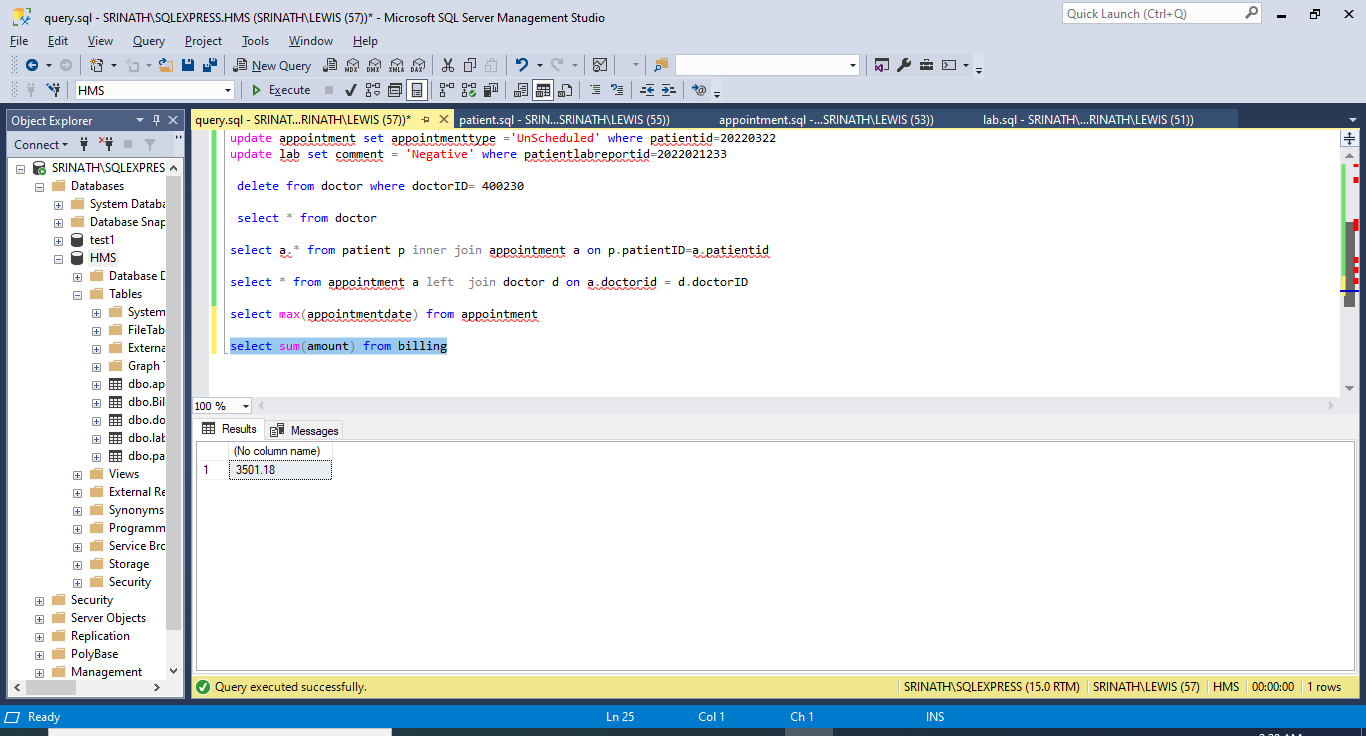


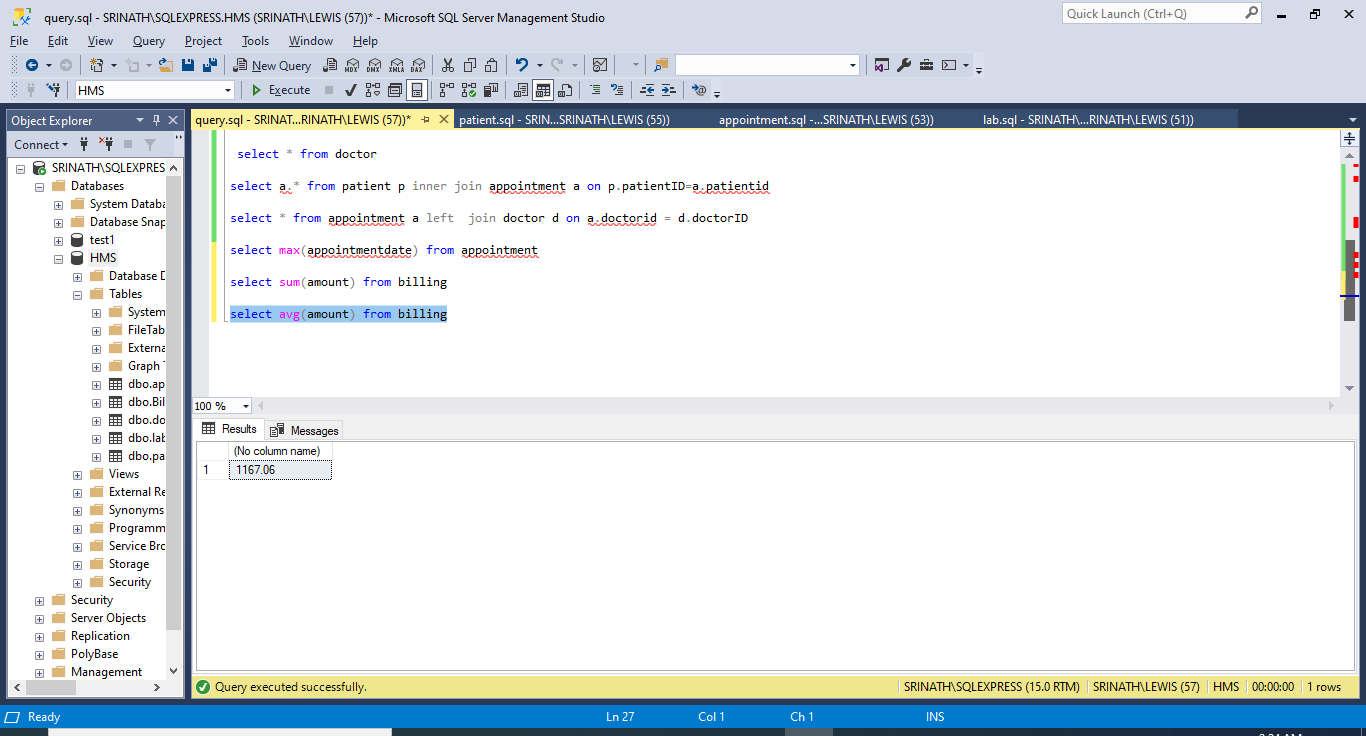


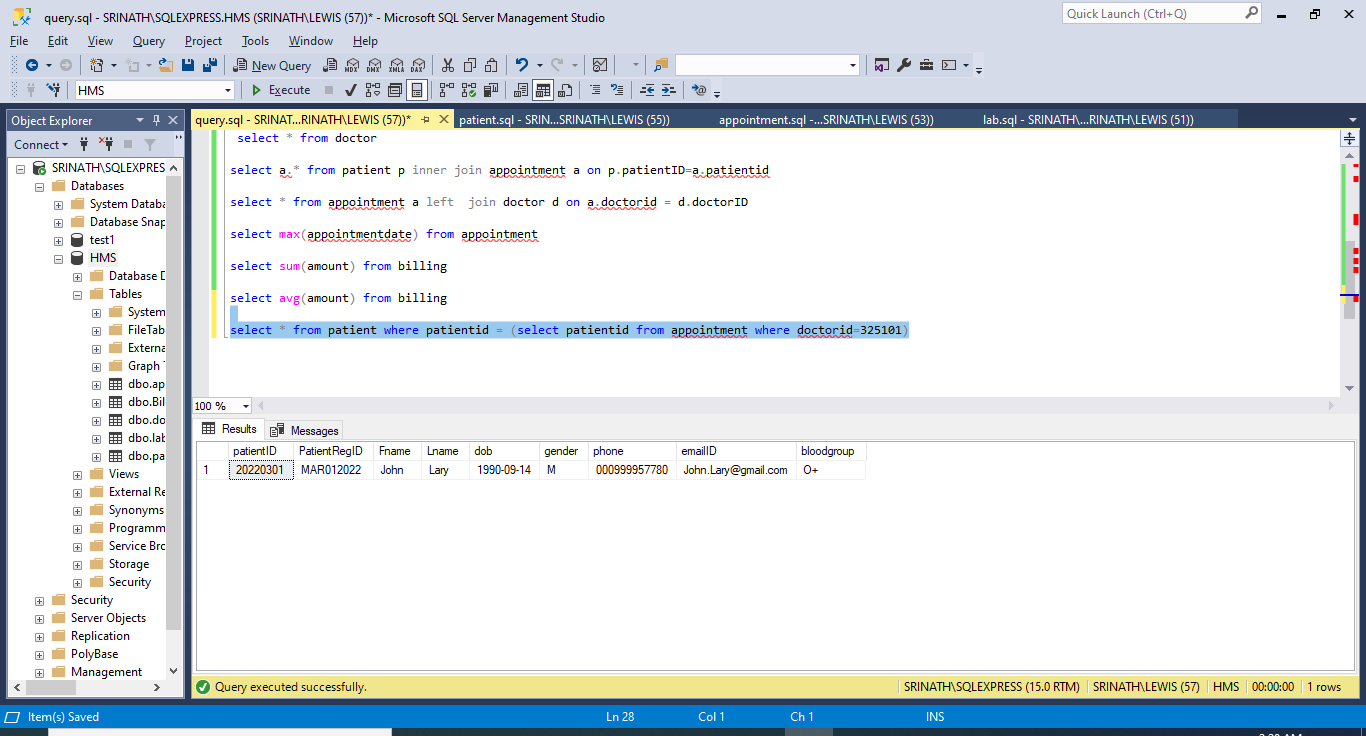












Project Part V:

Index:

In my project hospital management system, I am creating INDEXES on Patient and Billing tables.

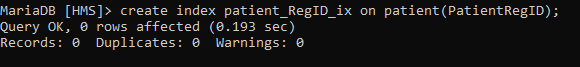
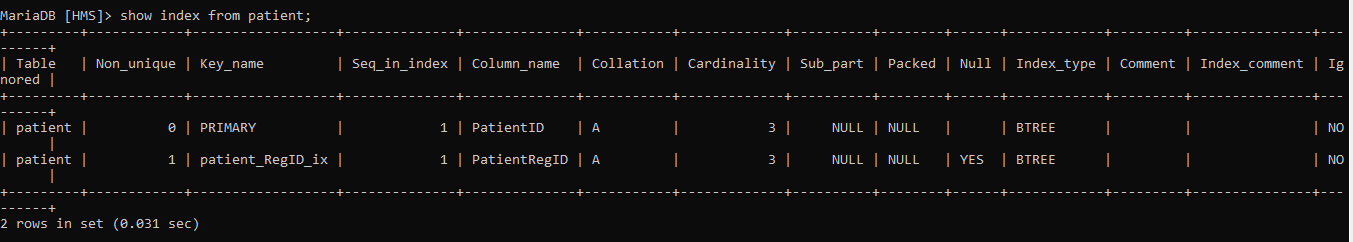
The ‘CREATE INDEX’ statement is used to create indexes in tables.

Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries.

First index is on Patient table. I am using ‘PatientRegID’ column because, it is most frequently used field at the hospital desk and the patient details should be retrieved quickly.

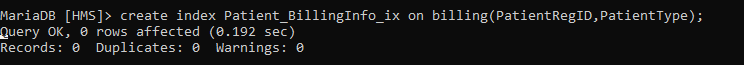
Syntax:

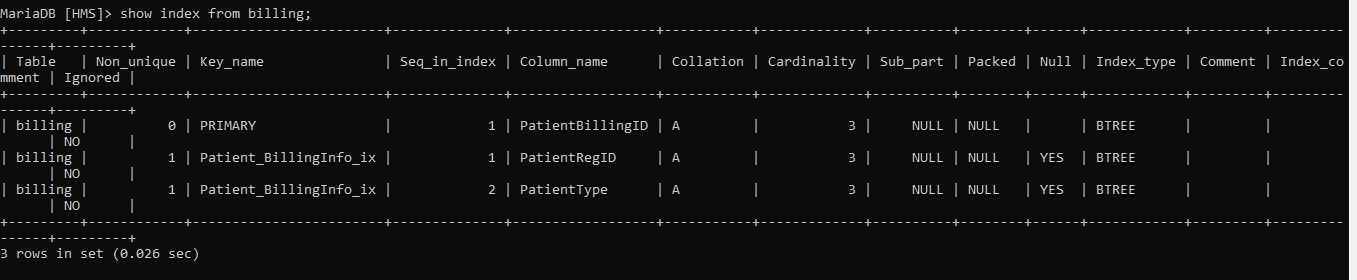
create index patient\_RegID\_ix on patient(PatientRegID);

The second index is on table ‘Billing’. Here I am using two columns ‘PatientRegID’,’PatientBillingType’.

I am creating this ‘composite’ index because these fields contain large number of distinct values. And also the details like payment mode (Cash/Insurance) can be retrieved and easily used by billing desk during discharge time.





There is a bunch of tools that we can use to do a performance test:

General purpose testing:

* PassMark PerformanceTest
* CPU-z
* CrystalDisk Mark
* Phoronix Test Suite

Database-related testing:

* Sqlio.exe
* HammerDb
* OStress

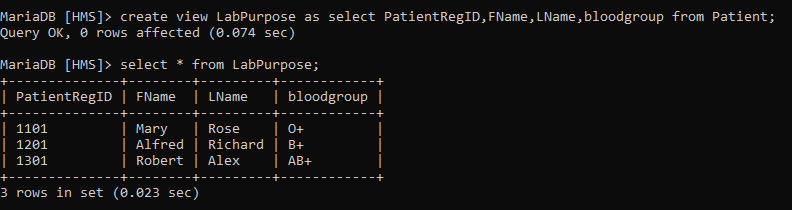
View:

View is a SELECT statement that's stored in the database as an object. It is also called a Virtual table. We can use a view anywhere we would normally use a table.

View1:

Syntax:

create view LabPurpose as select PatientRegID,FName,LName,bloodgroup from Patient;



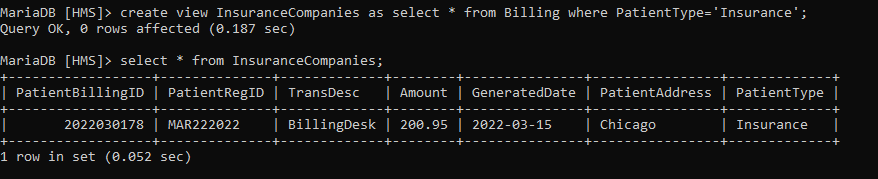
I want to share primary details of the patient from hospital to labs. So, I choose RegId,Name,Bloodgroup to form a view and the rest of the columns can’t be viewed by Lab people. In this way by using a view we can attain ‘Improved Security’ and ‘Convenience’.

View2:

There are situations where patient details should be segregated based on billing type. Either ‘cash’ or ‘Insurance’. And the patient details whose payment mode is ‘Insurance’ needs to be shared with Insurance groups and companies. Here comes the necessity for view2.

Syntax:

create view InsuranceCompanies as select \* from Billing where PatientType='Insurance';



By defining this view we can attain ‘Data Independence’ and ‘Reduced Complexity’.