Name : Darshan Puttanna
ID: 801169012

ITIS 6120 8120

Applied Databases-Project 2

Advanced Features for University Student Clinic Database System

Name : Darshan Puttanna
ID: 801169012

Introduction

EMR (Electronic Medical Records) system is a common and easy platform to expedite the functioning of a hospital on a large scale. Due to a large number of health issues in today's scenario, a platform that provides multilateral communication can save a lot of time. Our system gives considerate flexibility for doctors, patients and admin to access this system with ease. This platform provides fast end-to-end process completion from disease diagnosis to patient cure. It aims to improve visibility of all the procedures made in the case of a disease cure. This application will provide business intelligence data for better decision making of staffing, medicines, equipment's etc.

Our project is a database for an EMR system systematized collection of patient and population electronically stored health information in a digital format. These records can be shared across different health care settings. Generally, the EMR system is very complex, often consisting of a huge number of tables. In this project, we developed a database system that serves as the backend of a simple EMR system that captures a small portion of relevant information around a focused care scenario.

Scope

- EMR system is an integrated application for medical practices. Currently, the system plans to cover the general requirements of small unit. We limited our scope based on our domain knowledge and decided to focus on a dental clinic.
- Currently we have 3 user roles: Admin, Doctor, Pharmacist.
- Mentioned user roles are having access to specific tables as required by their role.
- As per role, users are restricted to perform action of other roles.
- System can be used to store appointment details.
- This system can be used to calculate billing details for patients.
- System can handle doctor schedules.
- This system can be used for patient profile creation.

Tools

We have used below tools in our project:

1. MySQL server on Workbench

Name : Darshan Puttanna
ID: 801169012

Requirement's specification

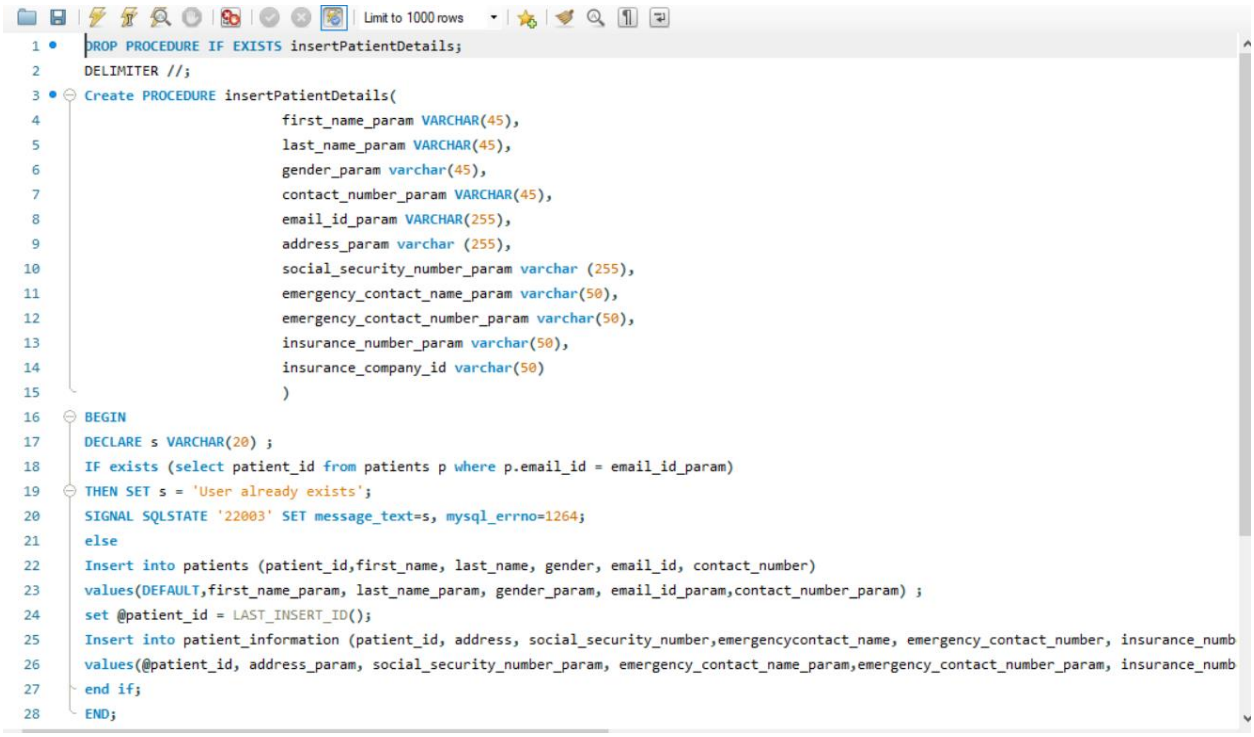
In Project 1, we have created the EMR database for University Student Clinic. We covered almost all requirements in the last project. In this phase, to improve database performance we created stored procedures, indexes, triggers, views and audit trail functionalities.

Stored Procedure:

A stored procedure is a prepared SQL queries that can be save and then just call it to execute It. Stored procedure can be reuse and hence those are efficient. If it contains parameters, it can act based on the parameter value(s) that is passed.

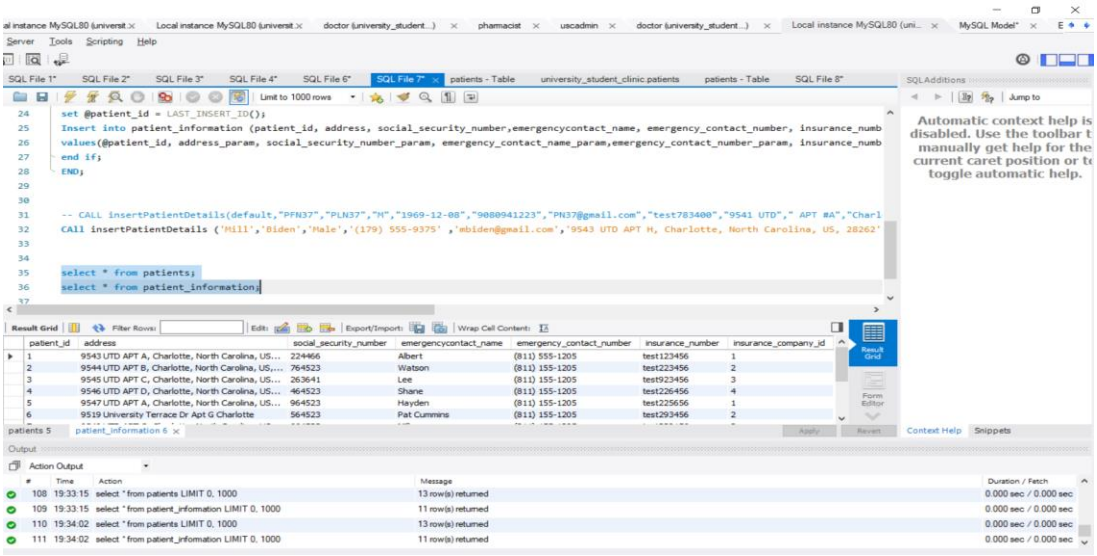
Requirement set 1:

- Create stored procedure to insert patient details in patient table.

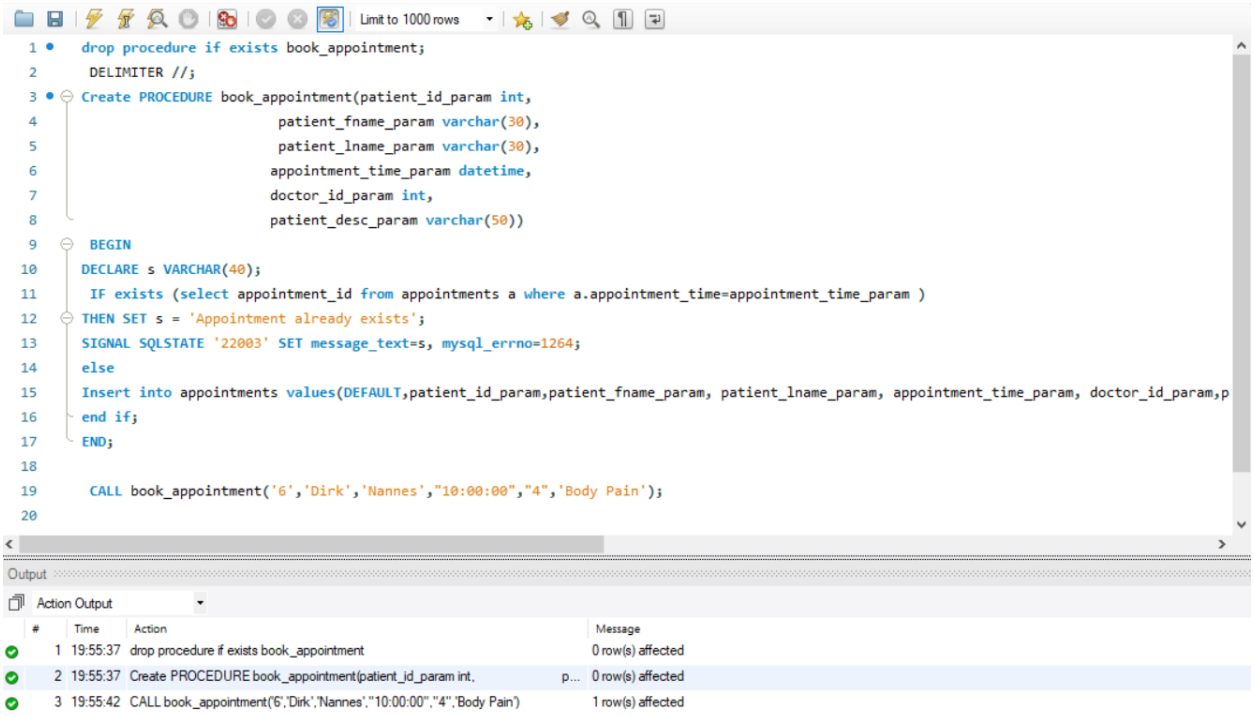


Name : Darshan Puttanna
ID: 801169012

Result: To show after calling the procedure patient details is inserted.

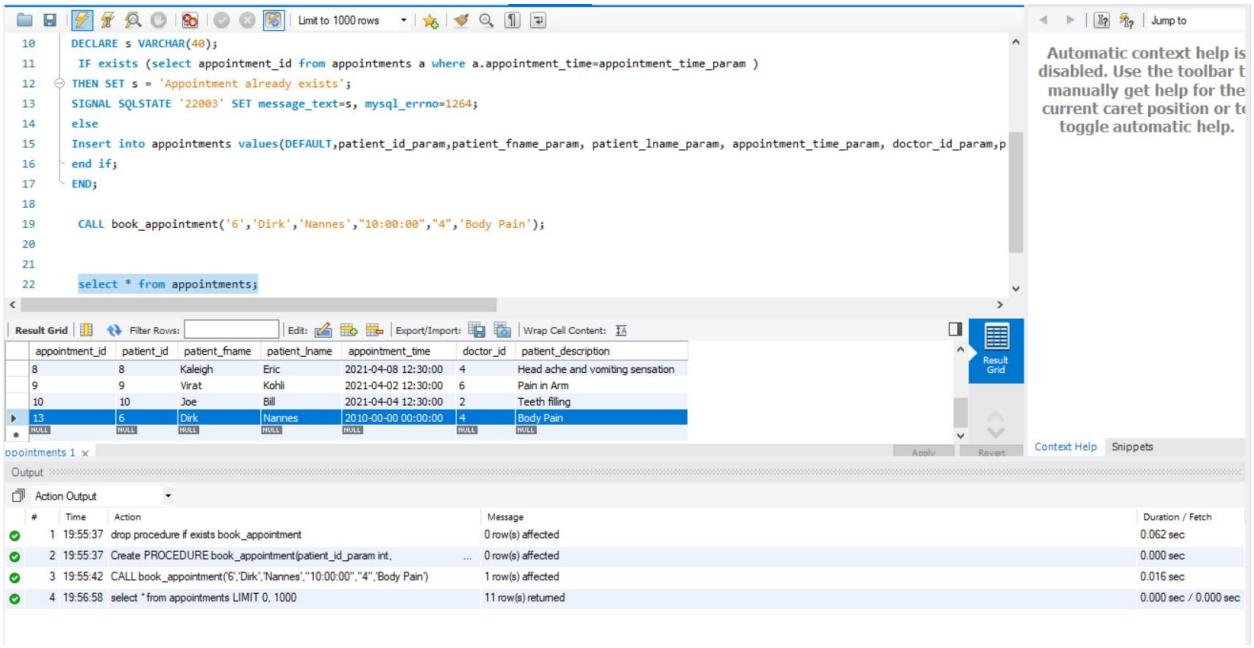


- Create stored procedure to insert appointment details in appointmentdetails table.

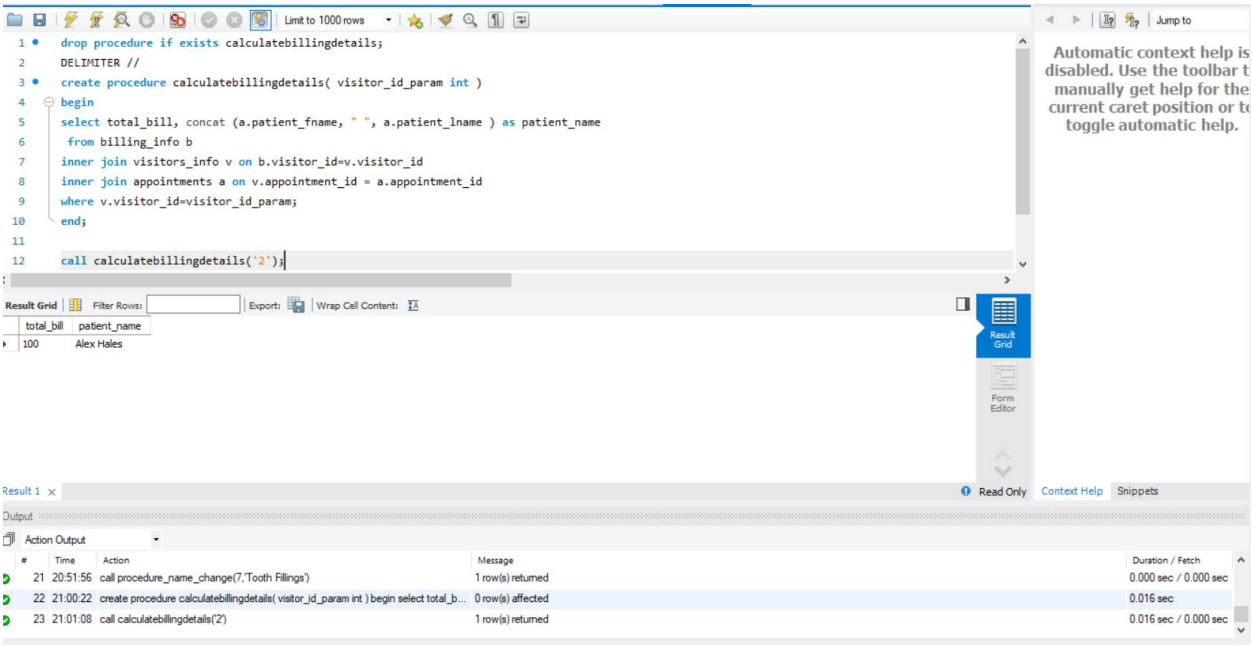


Name : Darshan Puttanna
ID: 801169012

We can see that patient successfully booked the appointment.



- Create stored procedure to display billing details for patients using vistor_id.

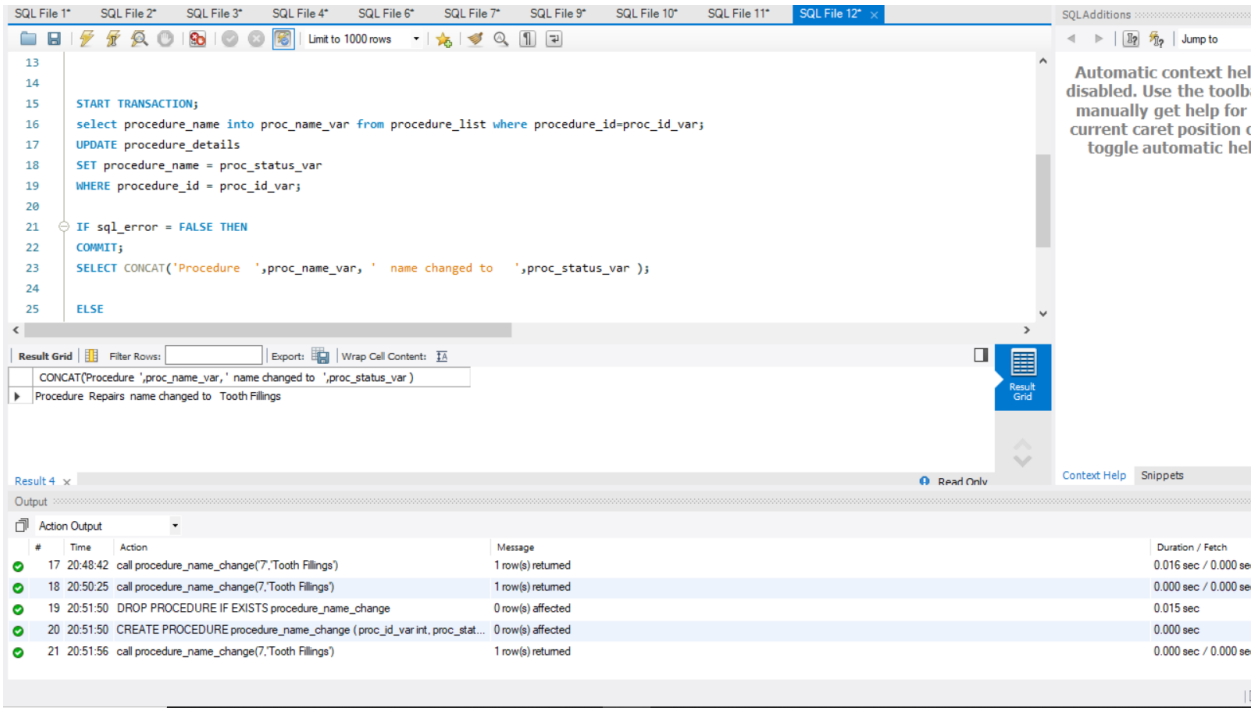


Name : Darshan Puttanna
ID: 801169012

- Create a stored procedure to update procedure list name.

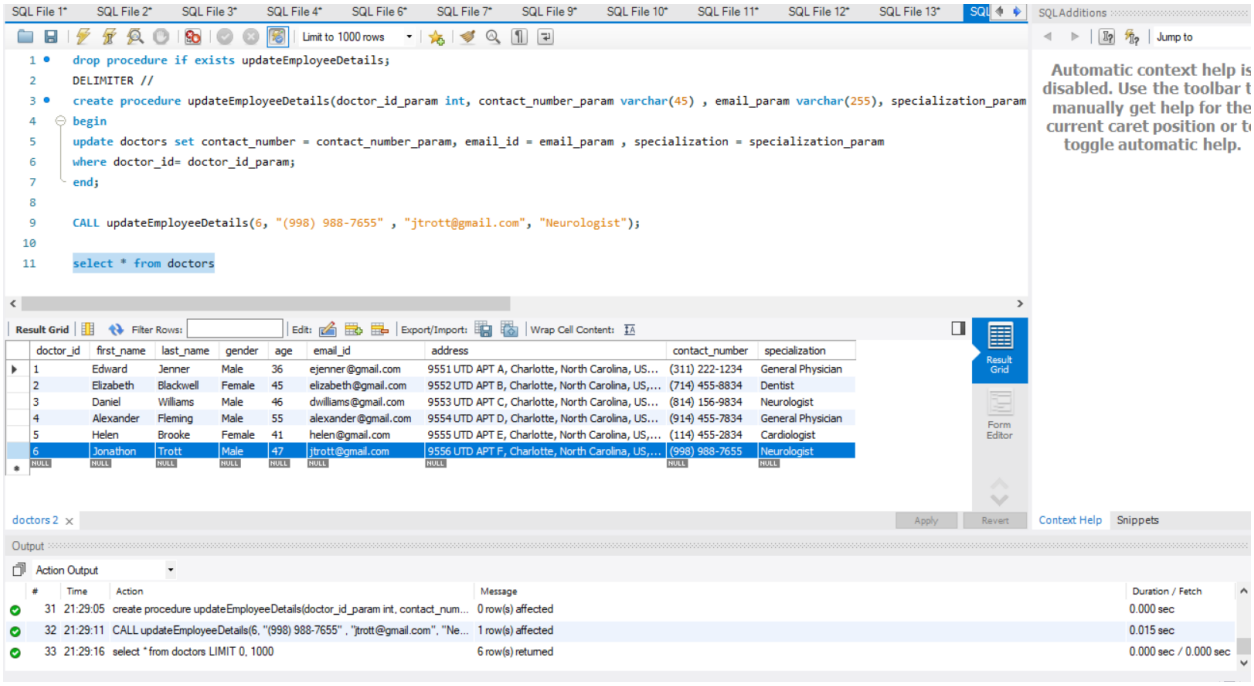
```
DROP PROCEDURE IF EXISTS procedure_name_change;
DELIMITER //
CREATE PROCEDURE procedure_name_change (
proc_id_var int,
proc_status_var varchar(50)
)
BEGIN
DECLARE sql_error tinytext DEFAULT FALSE;
declare proc_name_var varchar(50);
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION

START TRANSACTION;
select procedure_name into proc_name_var from procedure_list where procedure_id=proc_id_var;
UPDATE procedure_details
SET procedure_name = proc_status_var
WHERE procedure_id = proc_id_var;

IF sql_error = FALSE THEN
COMMIT;
SELECT CONCAT('Procedure  ',proc_name_var, '  name changed to   ',proc_status_var );

ELSE
ROLLBACK;
select ('Procedure name is not updated as error occured' );
END IF;
END//
DELIMITER ;
```

Result: Procedure name is updated.

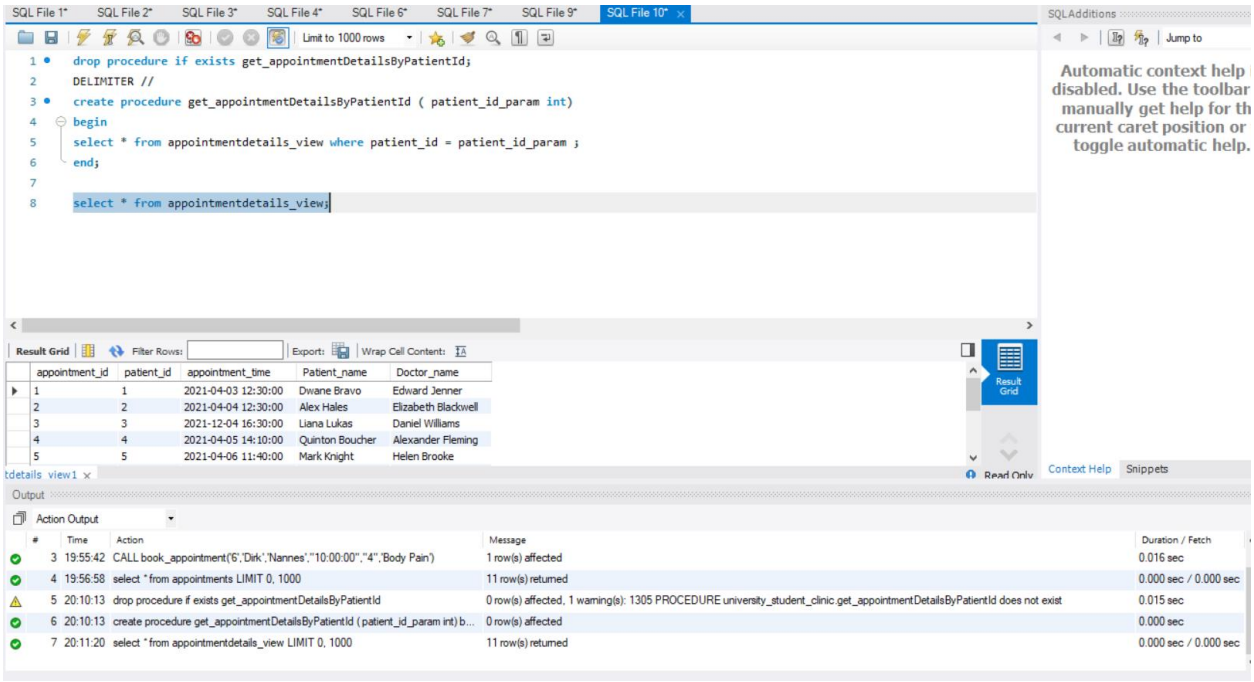


Name : Darshan Puttanna
ID: 801169012

- Create stored procedure to update employee(doctor) details.

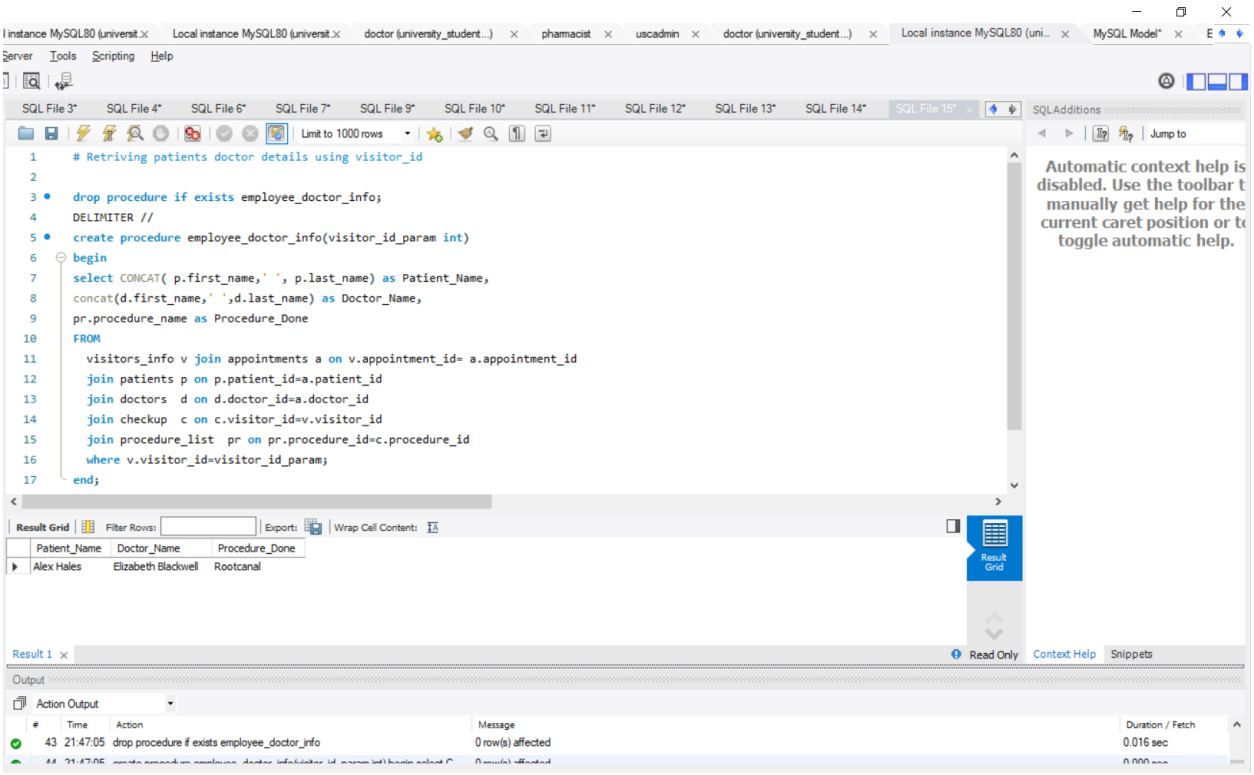


- Create stored procedure to display employee details for particular employee ID.



Name : Darshan Puttanna
ID: 801169012

- Create stored procedure to retrieve patient and consulted doctor details using visitor_id.

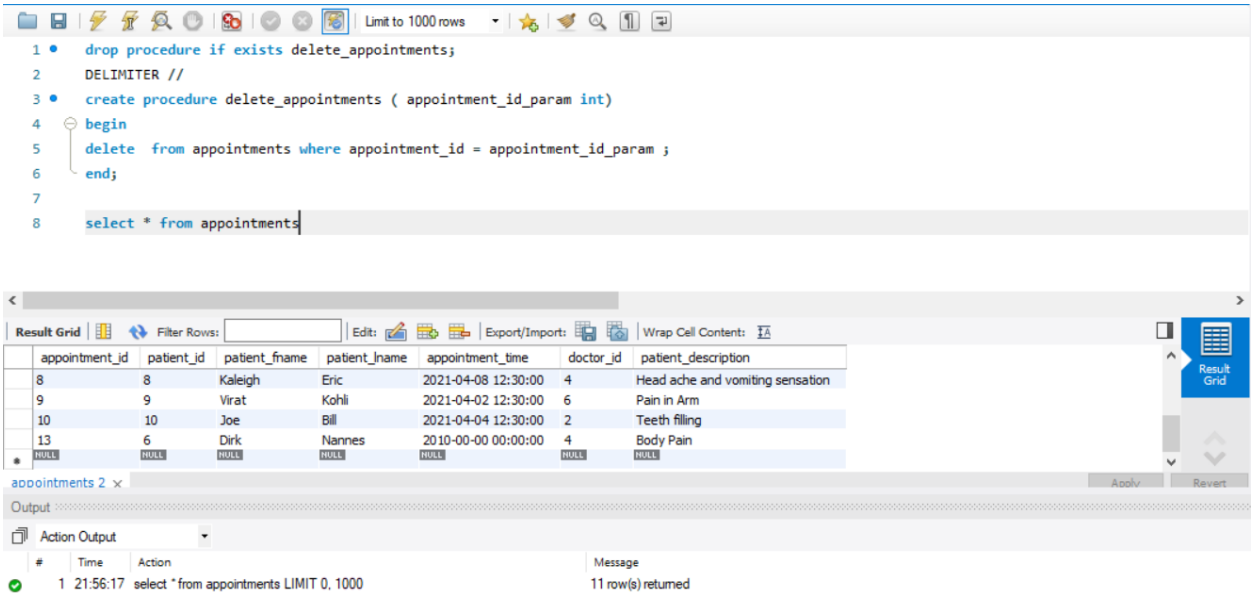


```
# Retriving patients doctor details using visitor_id

drop procedure if exists employee_doctor_info;
DELIMITER //
create procedure employee_doctor_info(visitor_id_param int)
begin
select CONCAT( p.first_name,' ', p.last_name) as Patient_Name,
concat(d.first_name,' ',d.last_name) as Doctor_Name,
pr.procedure_name as Procedure_Done
FROM
  visitors_info v join appointments a on v.appointment_id= a.appointment_id
  join patients p on p.patient_id=a.patient_id
  join doctors  d on d.doctor_id=a.doctor_id
  join checkup  c on c.visitor_id=v.visitor_id
  join procedure_list  pr on pr.procedure_id=c.procedure_id
  where v.visitor_id=visitor_id_param;
end;
```

Patient_Name	Doctor_Name	Procedure_Done
Alex Hales	Elizabeth Blackwell	Rootcanal

#	Time	Action	Message	Duration / Fetch
43	21:47:05	drop procedure if exists employee_doctor_info	0 row(s) affected	0.016 sec

- Create stored procedure to delete the appointment.



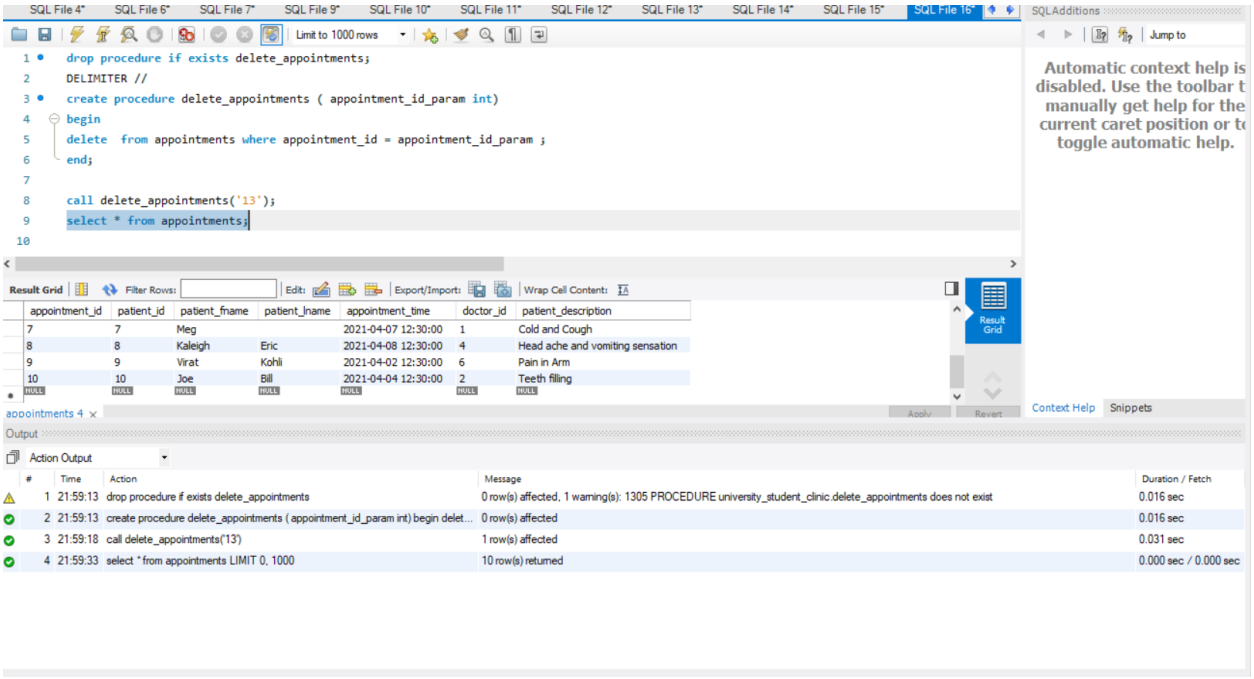
```
drop procedure if exists delete_appointments;
DELIMITER //
create procedure delete_appointments ( appointment_id_param int)
begin
delete  from appointments where appointment_id = appointment_id_param ;
end;

select * from appointments
```

appointment_id	patient_id	patient_fname	patient_lname	appointment_time	doctor_id	patient_description
8	8	Kaleigh	Eric	2021-04-08 12:30:00	4	Head ache and vomiting sensation
9	9	Virat	Kohli	2021-04-02 12:30:00	6	Pain in Arm
10	10	Joe	Bill	2021-04-04 12:30:00	2	Teeth filling
13	6	Dirk	Nannes	2010-00-00 00:00:00	4	Body Pain

#	Time	Action	Message
1	21:56:17	select * from appointments LIMIT 0, 1000	11 row(s) returned

Name : Darshan Puttanna
ID: 801169012

Form the below screenshot we can see that appointments with appointment_id = 13;



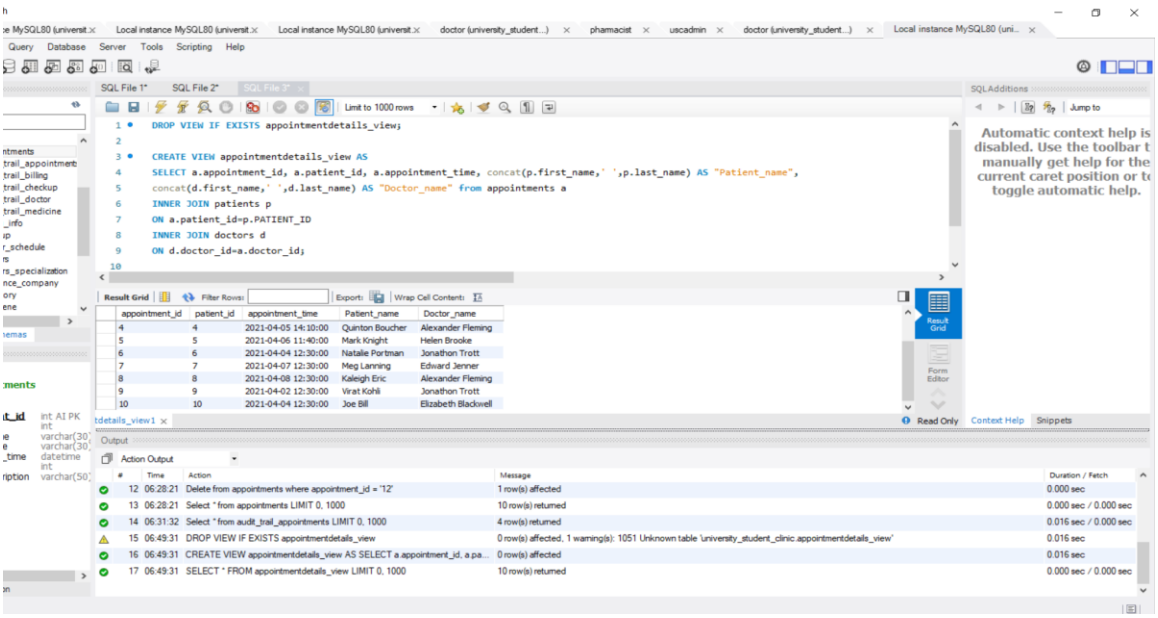
Name : Darshan Puttanna
ID: 801169012

Views:

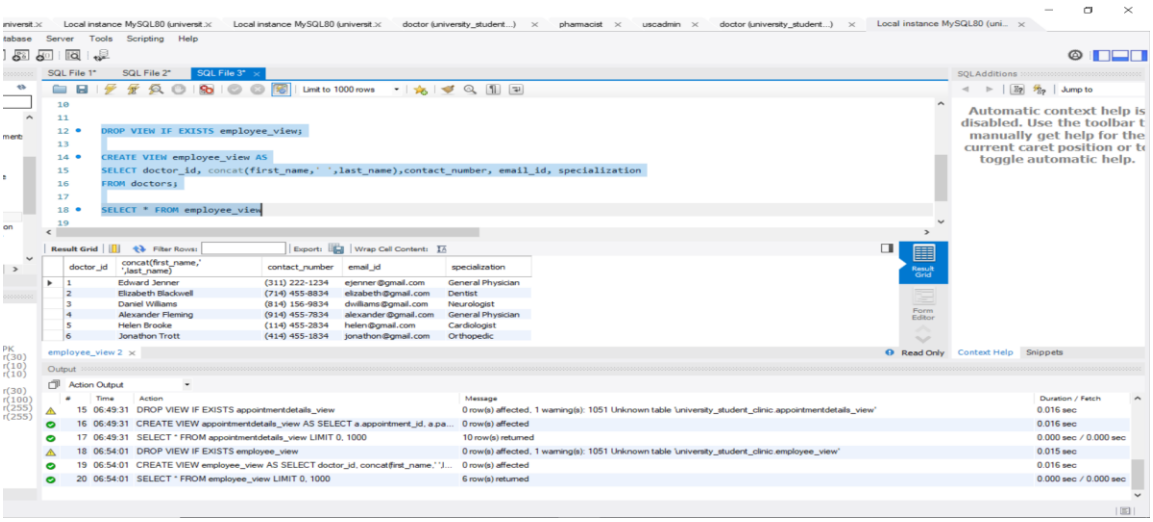
A view is a virtual table based on the result-set of an SQL queries. It contains rows and columns. Columns in view are the columns from one or more real tables in the database. We can add WHERE and JOIN statements to a view and present the data as if the data were coming from one single table.

Requirement set 2:

- Create a view to display appointment details.

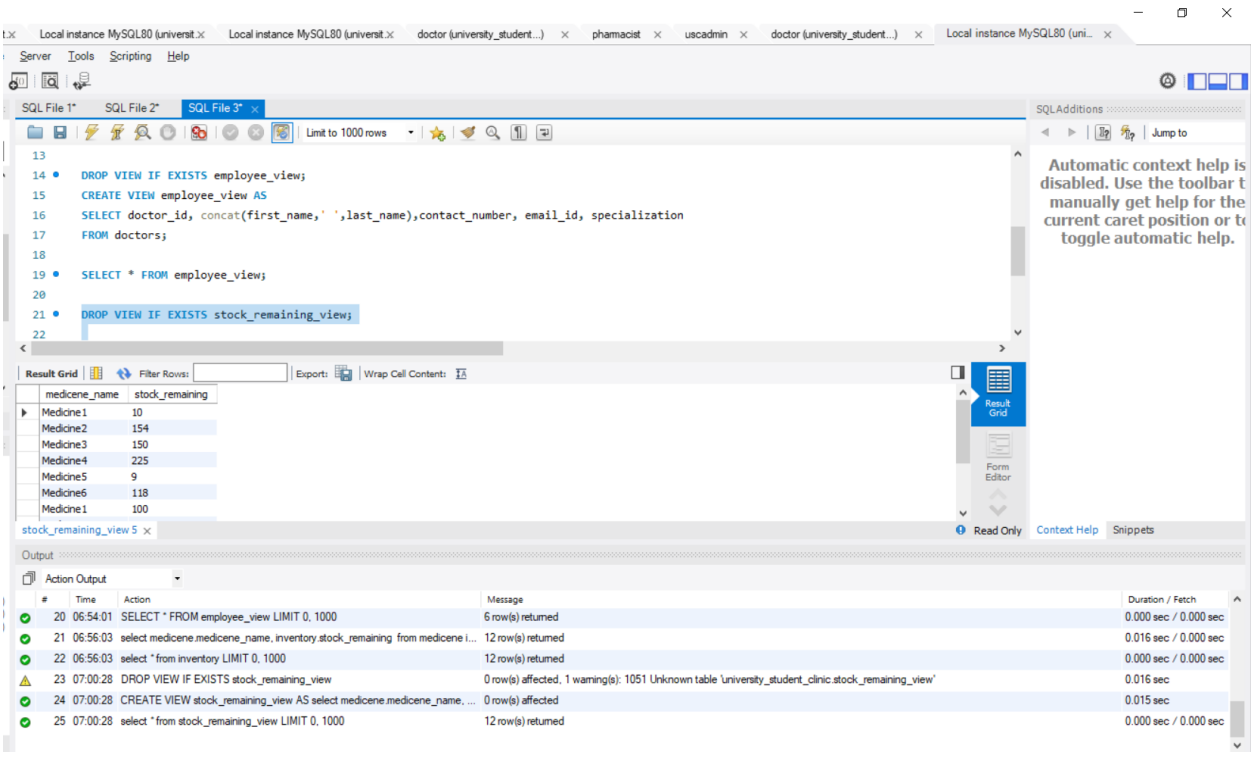


- Create a view to display doctor details.

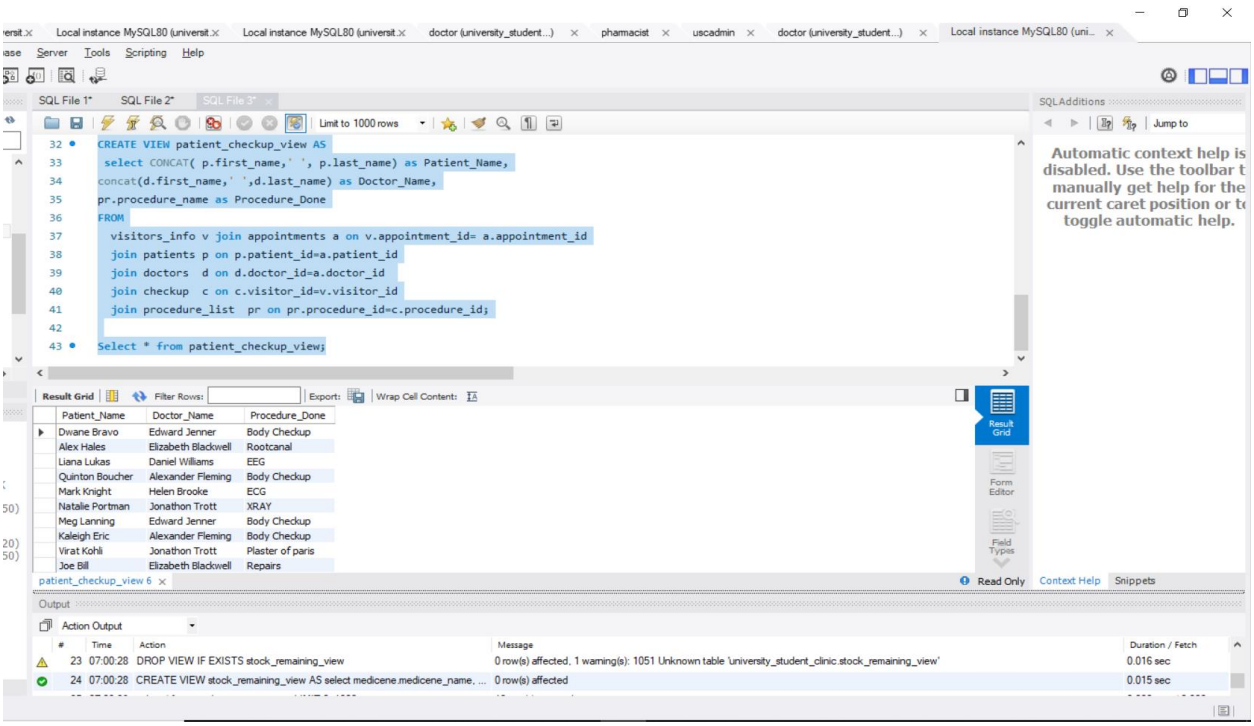


Name : Darshan Puttanna
ID: 801169012

- Create a view to display medicine stock in inventory.

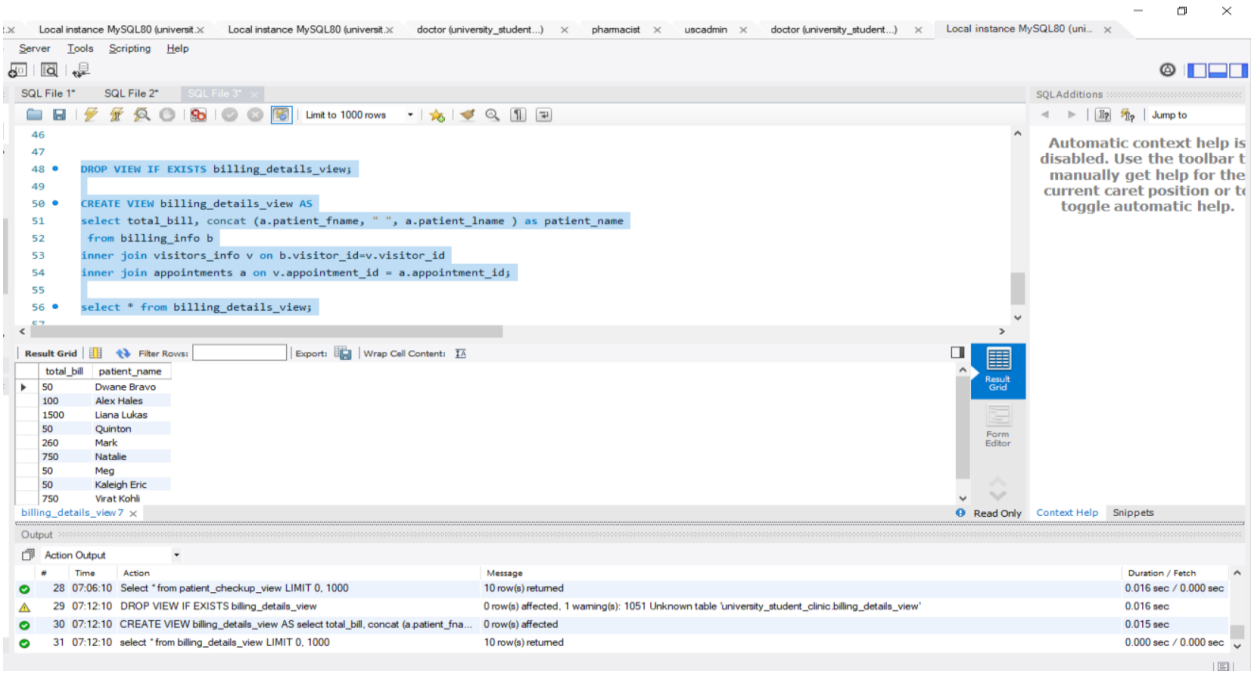


- Create a view to display patient's and consulted doctor as patient_checkup_view.



Name : Darshan Puttanna
ID: 801169012

- Create a view to display billing details.



Name : Darshan Puttanna
ID: 801169012

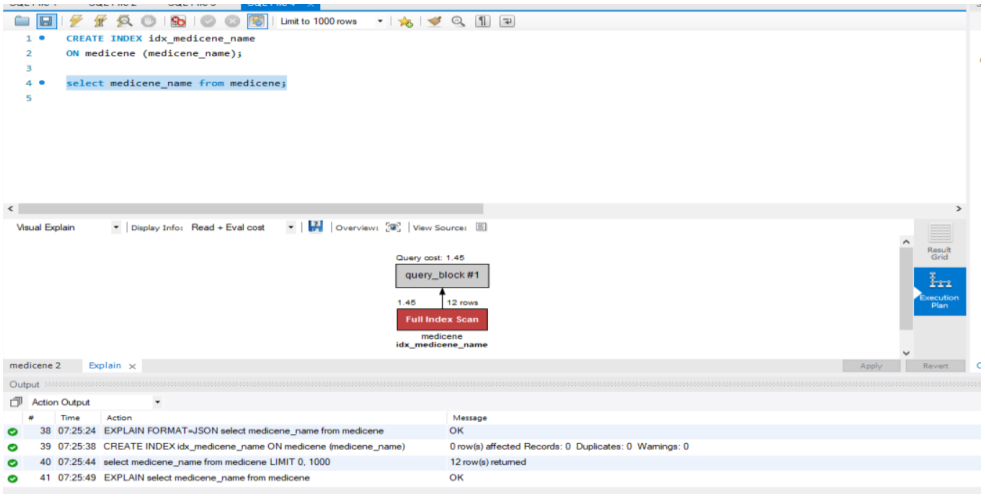
Index:

These are used to retrieve data from the database more quickly than usual. The users cannot see the indexes but indexes are just used to speed up searches/queries.

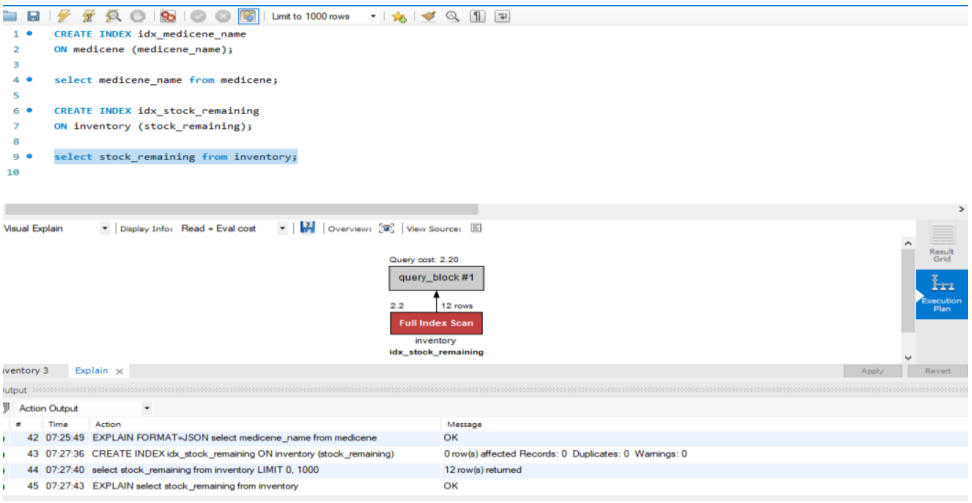
Requirement set 3:

We created the index for columns which we are frequently using to fetch the data necessary for the requirements of our project.

- Create index for medicine name column of medicine table.

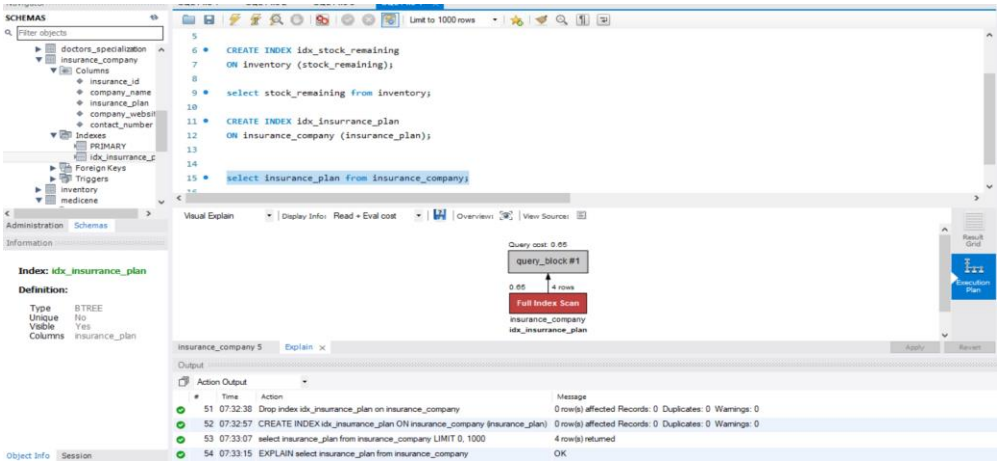


- Create index for stock remaining column of inventory table.

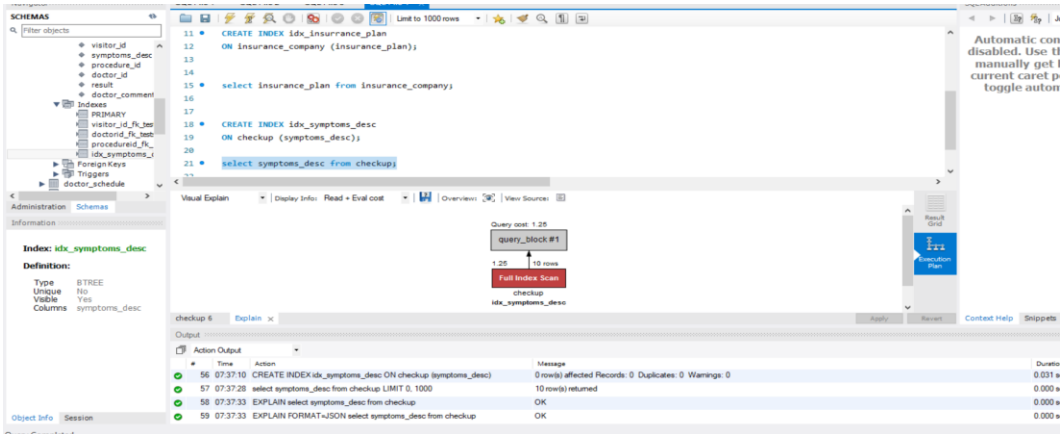


Name : Darshan Puttanna
ID: 801169012

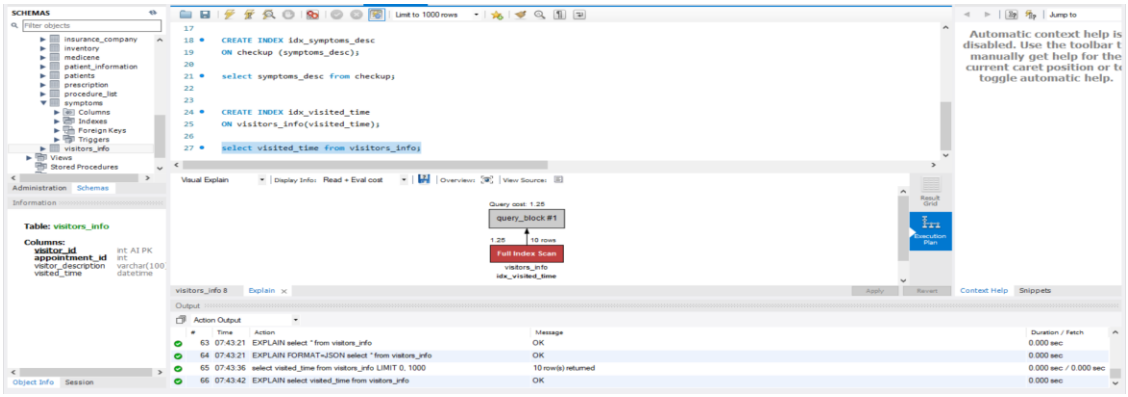
- Create index for insurance company column of insurance company table.



- Create index for symptoms description column of checkup table.



- Create index for visit date column of visit_information table.



Name : Darshan Puttanna
ID: 801169012

Authentication for different roles:

We have 3 roles in our project:

Requirement set 4:

Create 3 users and grant them access for different operation as required.

```
DROP USER IF EXISTS 'uscadmin'@'localhost' ;
DROP USER IF EXISTS 'doctor'@'localhost';
DROP USER IF EXISTS 'pharmacist'@'localhost';


CREATE USER 'uscadmin'@'localhost' IDENTIFIED BY 'uscadmin123456';
CREATE USER 'doctor'@'localhost' IDENTIFIED BY 'doctor123456';
CREATE USER 'pharmacist'@'localhost' IDENTIFIED BY 'pharmacist123456';

set password for 'uscadmin'@'localhost' = '123456';
set password for 'doctor'@'localhost' = '123456';
set password for 'pharmacist'@'localhost' = '123456';
```

1. Admin:

   Access: Admin has access for all the tables

```
GRANT ALL ON university_student_clinic.* TO 'uscadmin'@'localhost';
```

   Views: appointmentdetails_view, employee_view

2. Doctor

   Read access:

   Tables: medicine, appointments, billing_info, medicine, patients, procedure_list, symptoms

   Views: appointmentdetails_view, employee_view,

Name : Darshan Puttanna
ID: 801169012

Read and write access:

```
GRANT SELECT ON university_student_clinic.appointments TO 'doctor'@'localhost';
GRANT SELECT ON university_student_clinic.billing_info TO 'doctor'@'localhost';
GRANT SELECT,INSERT, UPDATE ON university_student_clinic.checkup TO 'doctor'@'localhost';
GRANT SELECT,INSERT, UPDATE ON university_student_clinic.doctors TO 'doctor'@'localhost';
GRANT SELECT,INSERT, UPDATE ON university_student_clinic.doctor_schedule TO 'doctor'@'localhost';
GRANT SELECT ON university_student_clinic.medicene TO 'doctor'@'localhost';
GRANT SELECT ON university_student_clinic.patients TO 'doctor'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON university_student_clinic.prescription TO 'doctor'@'localhost';
GRANT SELECT ON university_student_clinic.procedure_list TO 'doctor'@'localhost';
GRANT SELECT ON university_student_clinic.symptoms TO 'doctor'@'localhost';
```

3. Pharmacist

Read access:

Tables: checkup

Views: stock_remaining_view, patient_checkup_view

Read and write access:

Tables: inventory, medicine

```
GRANT SELECT,INSERT, UPDATE, DELETE ON university_student_clinic.inventory TO  'pharmacist'@'localhost';
GRANT SELECT,INSERT, UPDATE, DELETE ON university_student_clinic.medicene TO  'pharmacist'@'localhost';
GRANT SELECT ON university_student_clinic.checkup TO  'pharmacist'@'localhost';
```

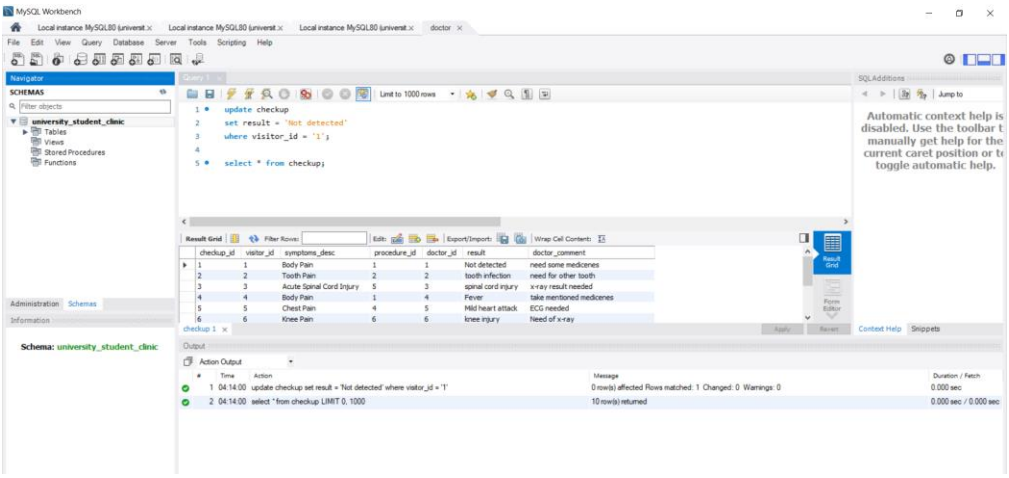
Name : Darshan Puttanna
ID: 801169012

Role based scenarios:

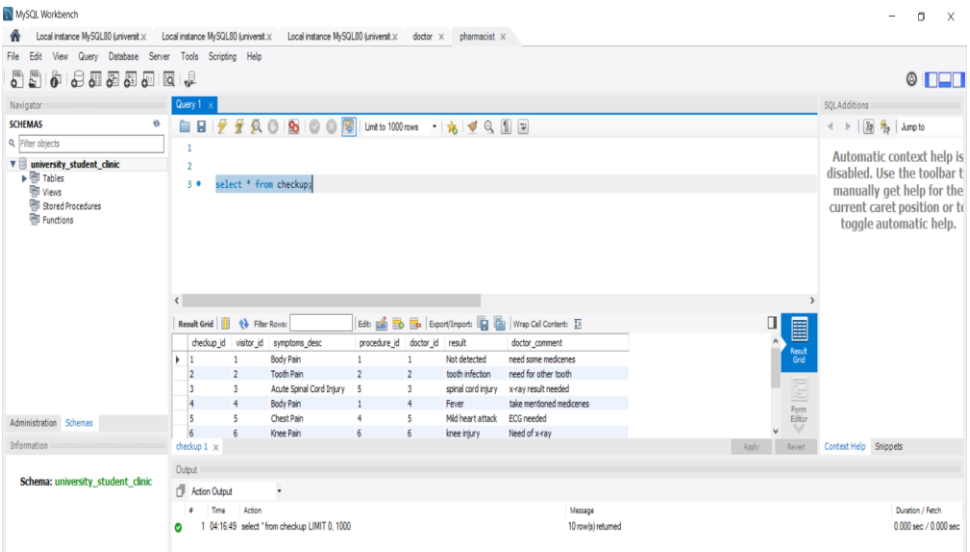
Requirement set 5:

1. Doctor can update the checkup details but pharmacist can only view those details. They are not allowed to update it.

   Updated for doctor role.

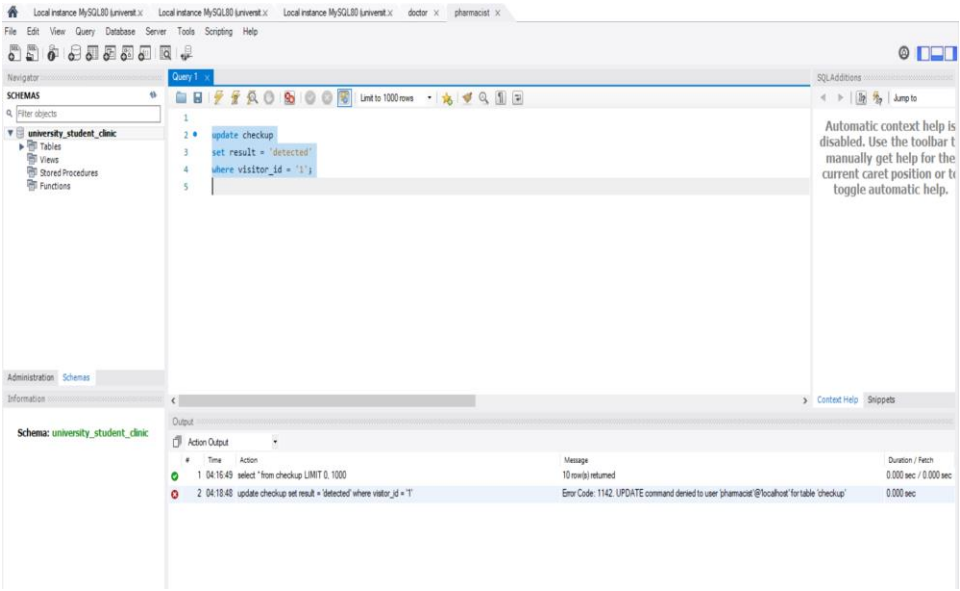


   Only pharmacist can view.

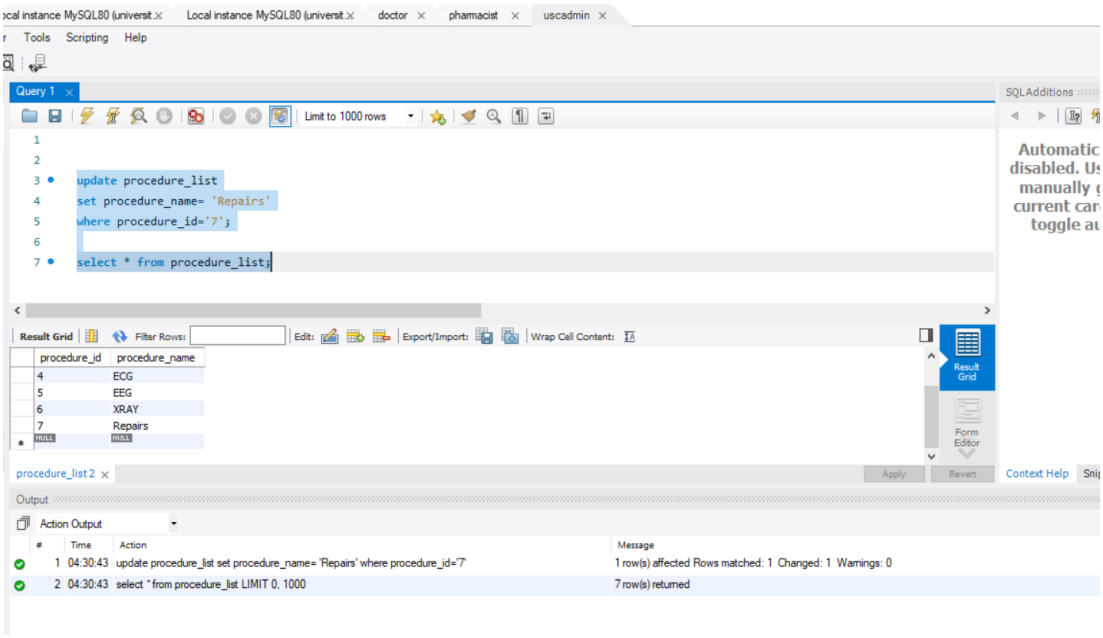


Name : Darshan Puttanna
ID: 801169012

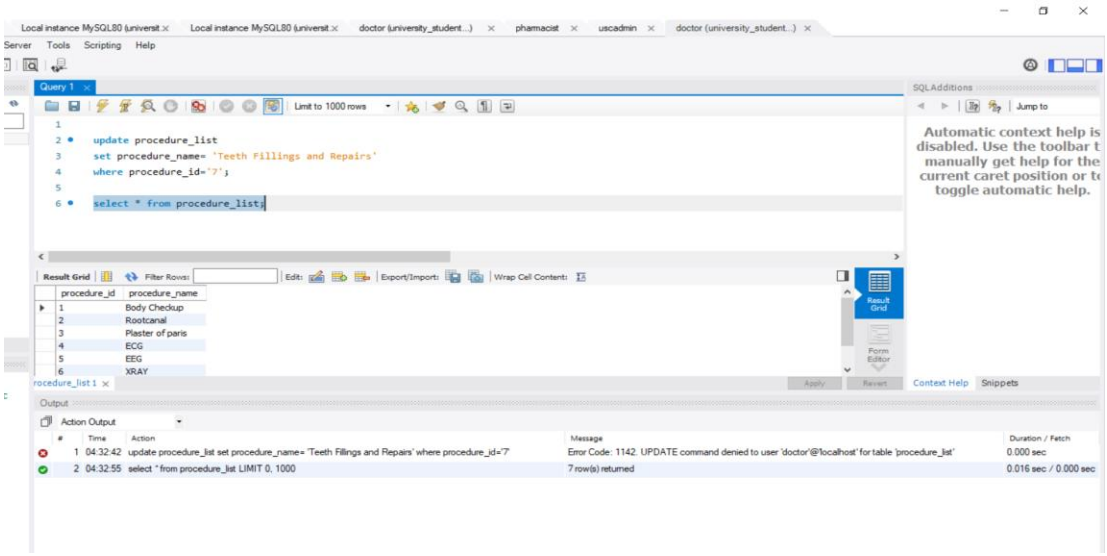
But cannot update.



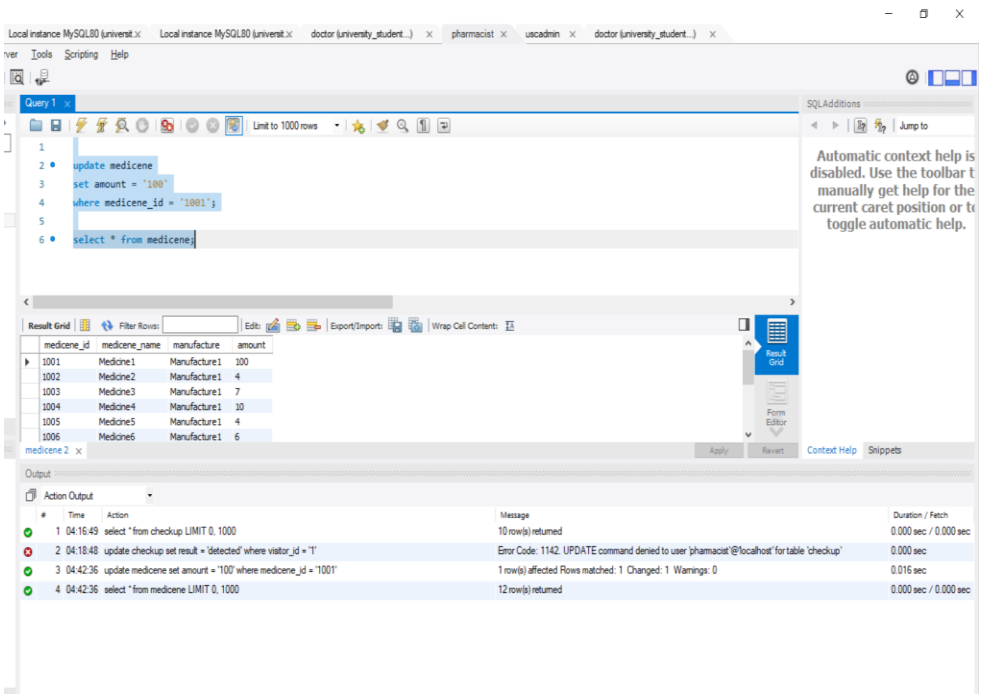
2. Admin can update procedure_list but doctor can only view the procedure_list.



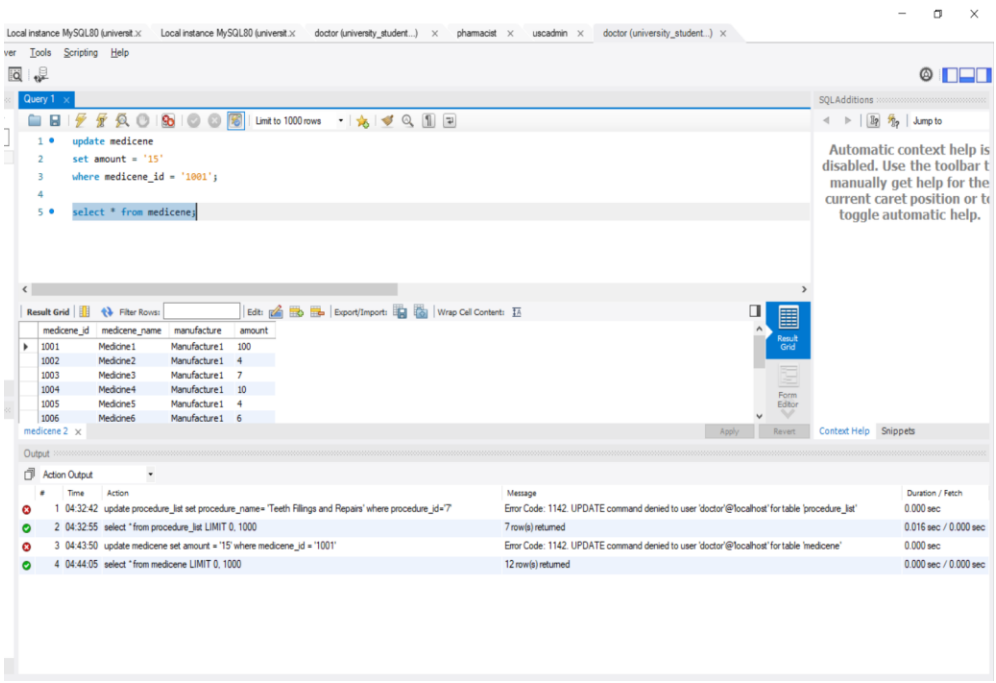
Name : Darshan Puttanna
ID: 801169012



3. Pharmacist can update the medicine details but doctor will have only view access to it.



Name : Darshan Puttanna
ID: 801169012



Audit trail:

We have added some audit tables to track the history of the data when it is accessed by any of the users. The tables will store the user who have accessed the data and the timing at which he accessed the data.

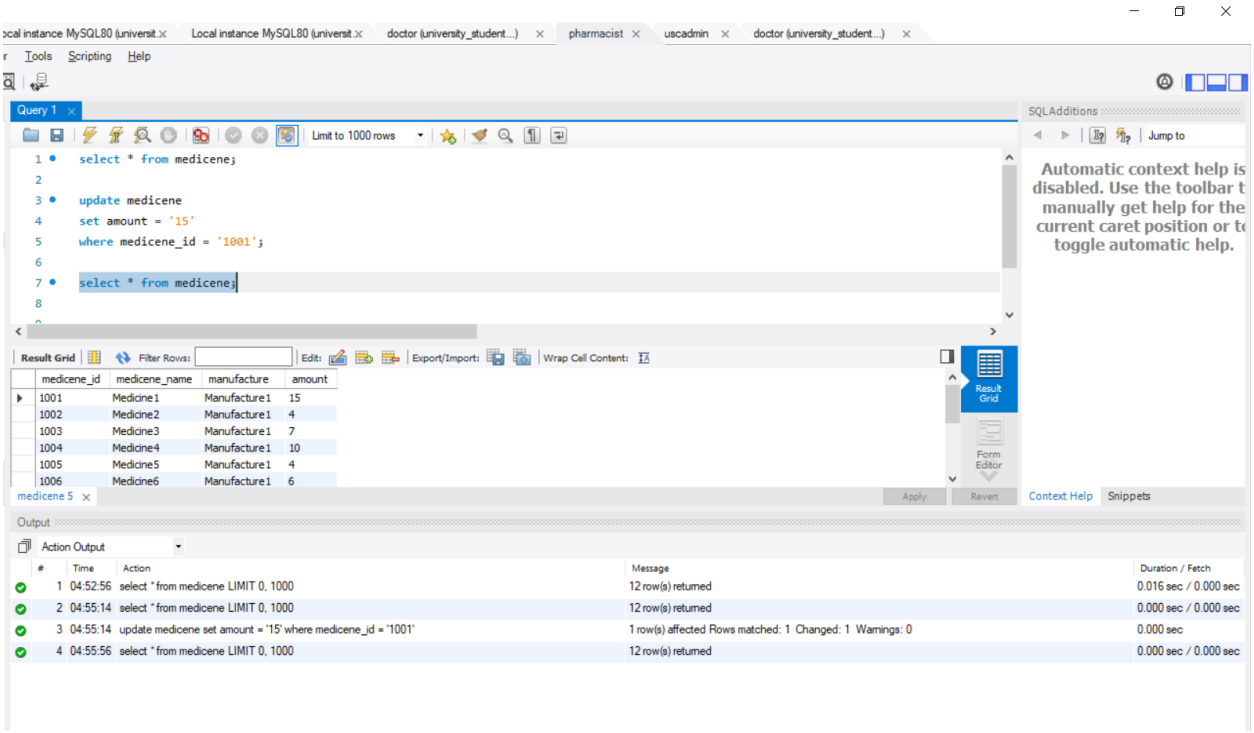
Name : Darshan Puttanna
ID: 801169012

Requirement set 6:

Scenarios:

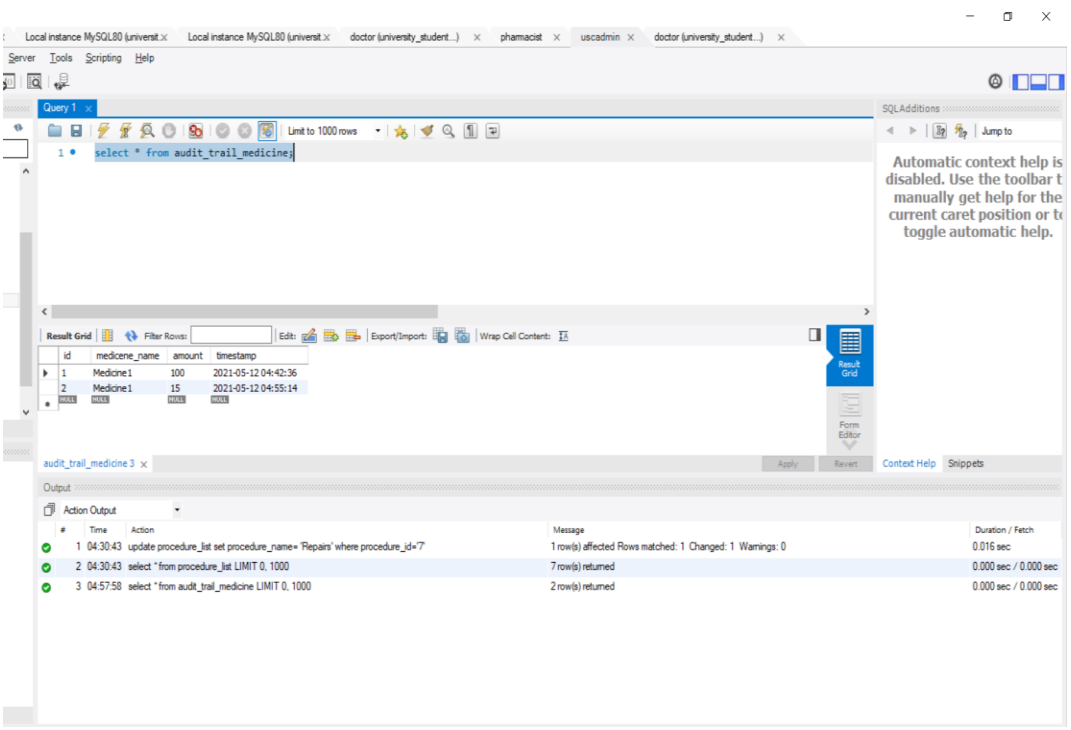
Insert data in audit table when user perform update action on the data.

1. When pharmacist update medicine amount in medicine table, the old value of medicine should get updated in audit history with timestamp and this can be performed by pharmacist.

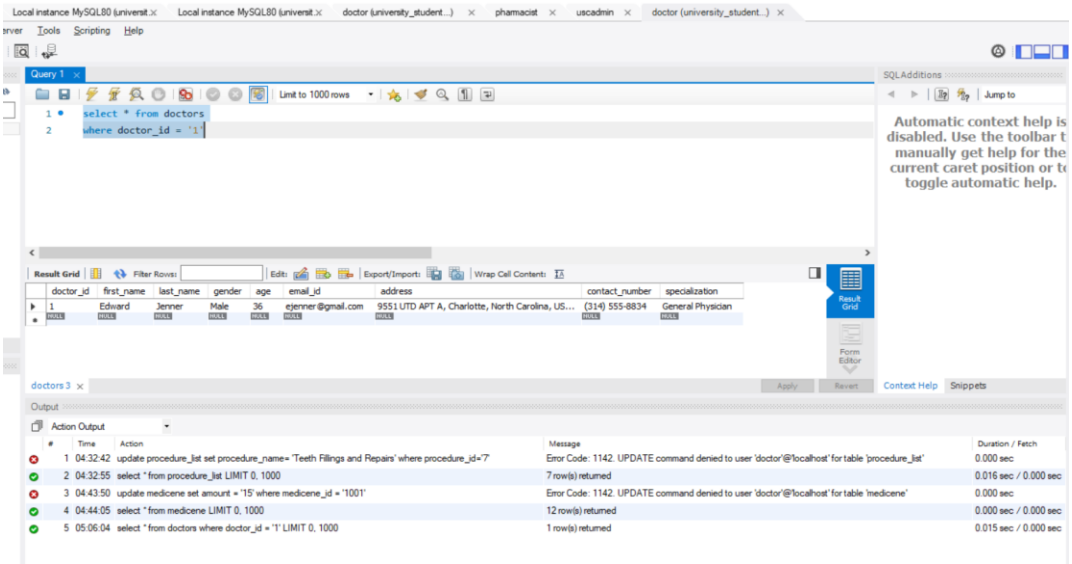


After the update by pharmacist, if we check the audit_trail_medicene by admin we can see the chages. This shows that trigger statement is working.

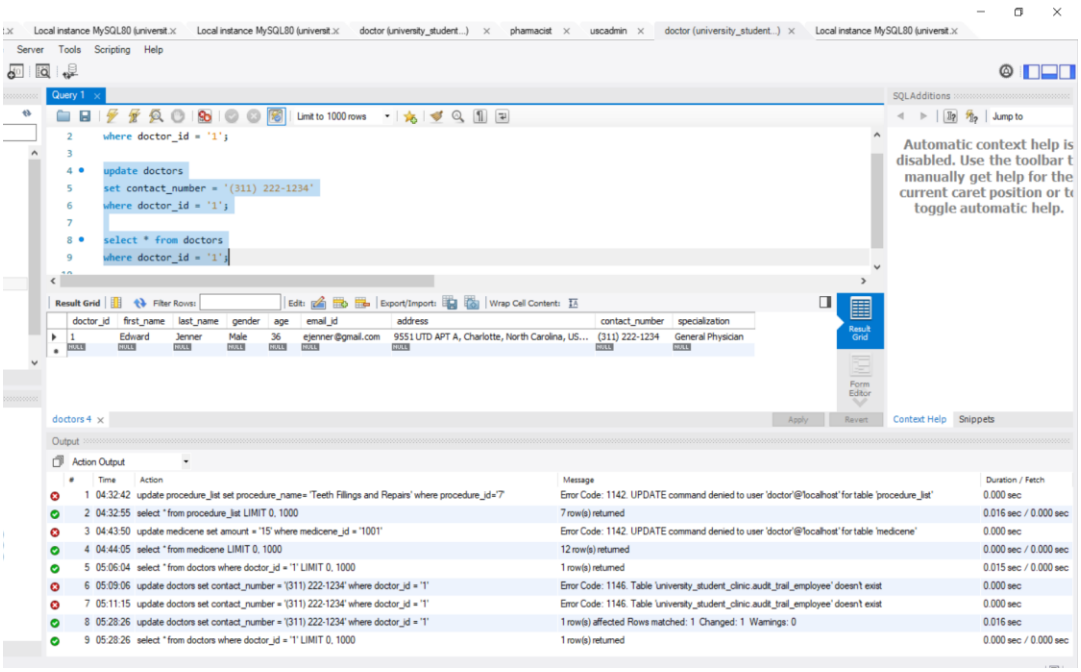
Name : Darshan Puttanna
ID: 801169012



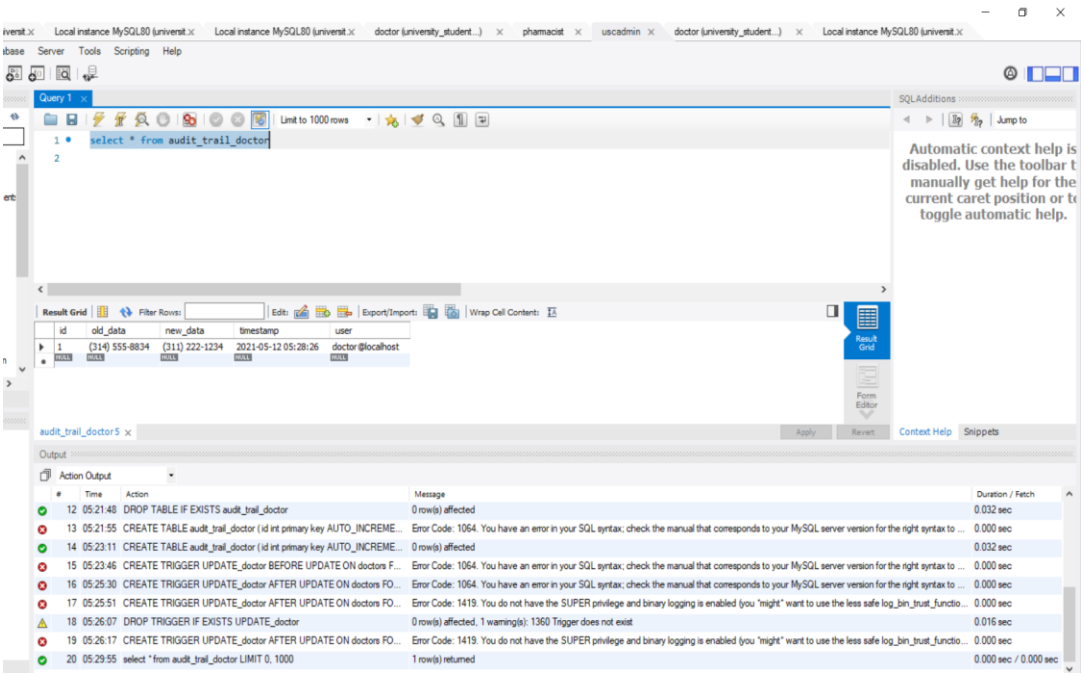
2. When doctor(employee) update their phone number in doctor table, the old value should get updated in audit history with timestamp.



Name : Darshan Puttanna
ID: 801169012



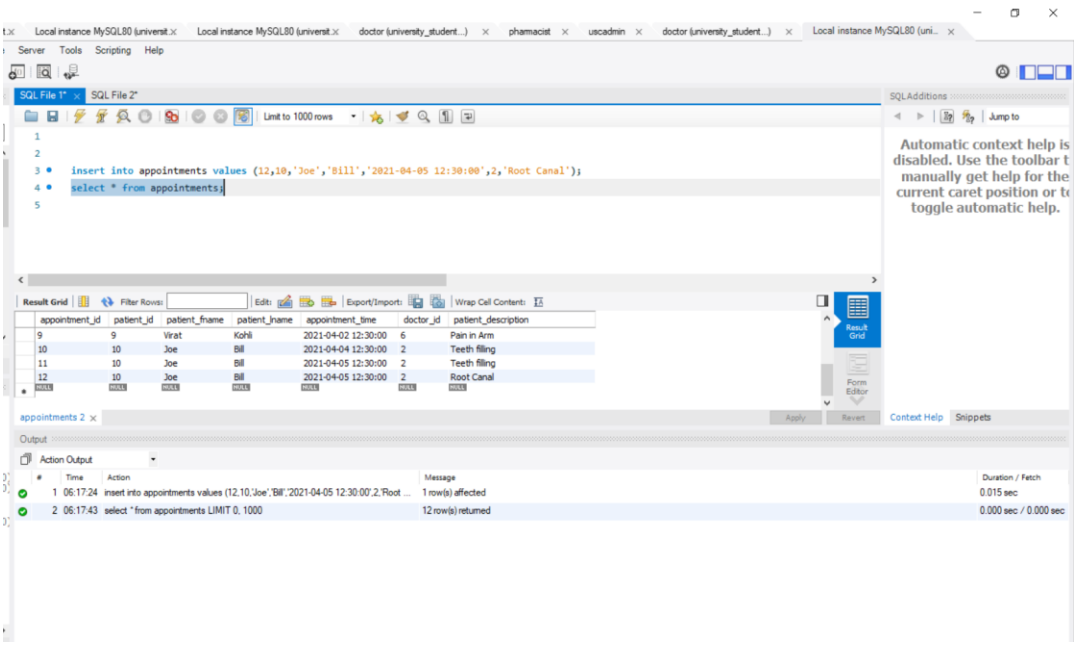
After doctor update table



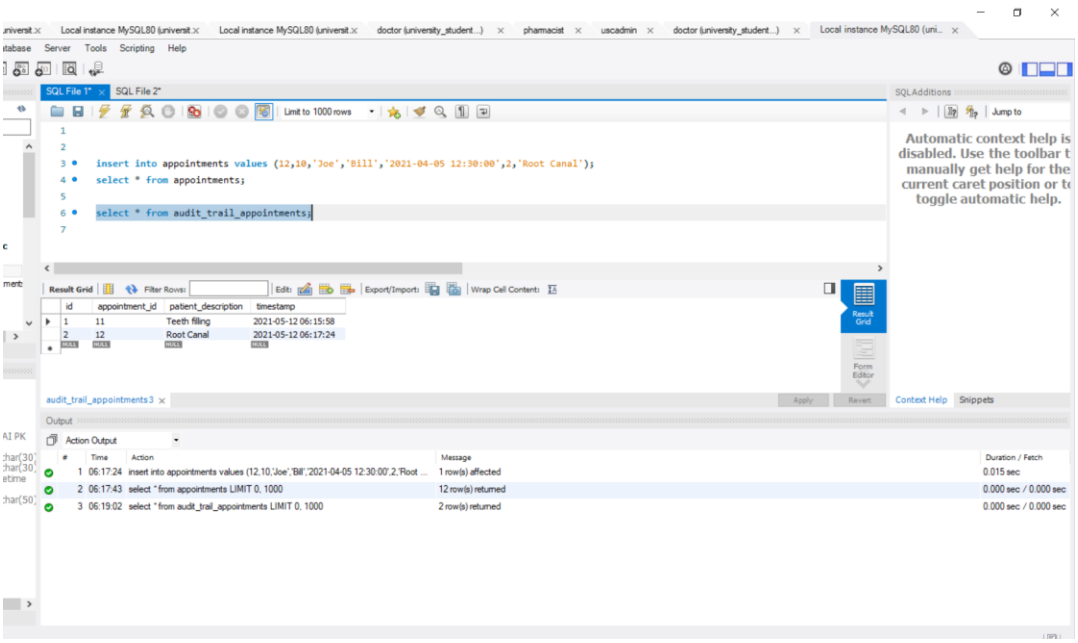
Name : Darshan Puttanna
ID: 801169012

Insert data in audit table when user perform insert action on data.

1. When patient book an appointment, audit table should get updated with their appointment_id and timestamp at which they are booking the appointment.



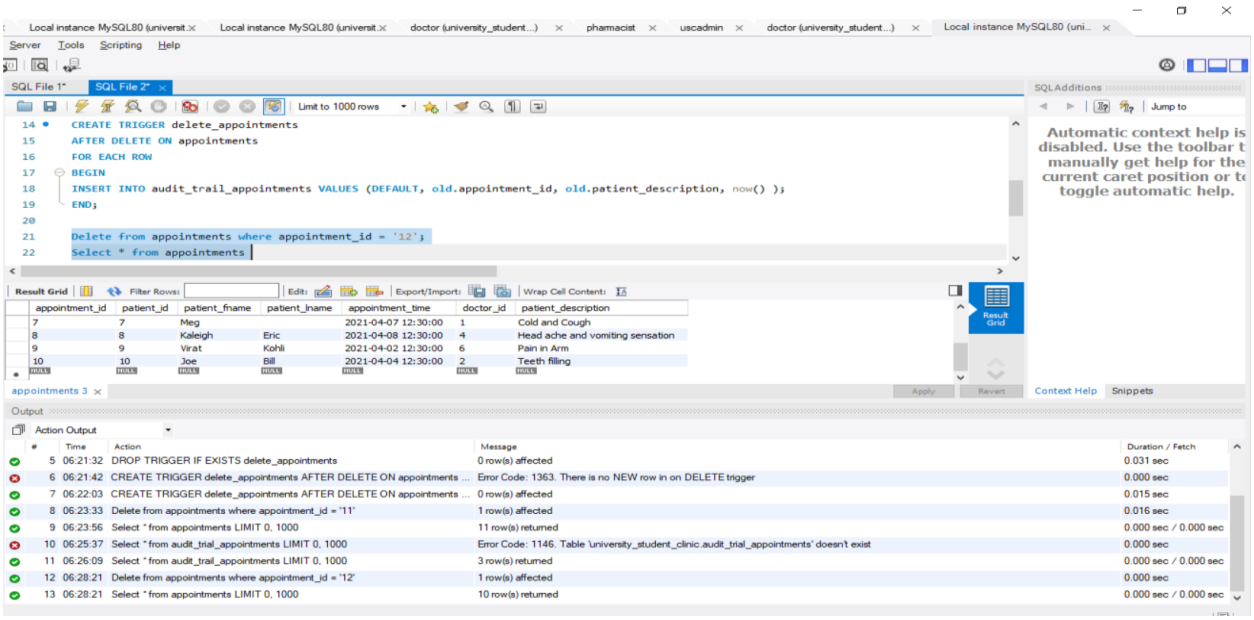
After inserting the data when we ran audit_trial_appointments the value is getting updated.



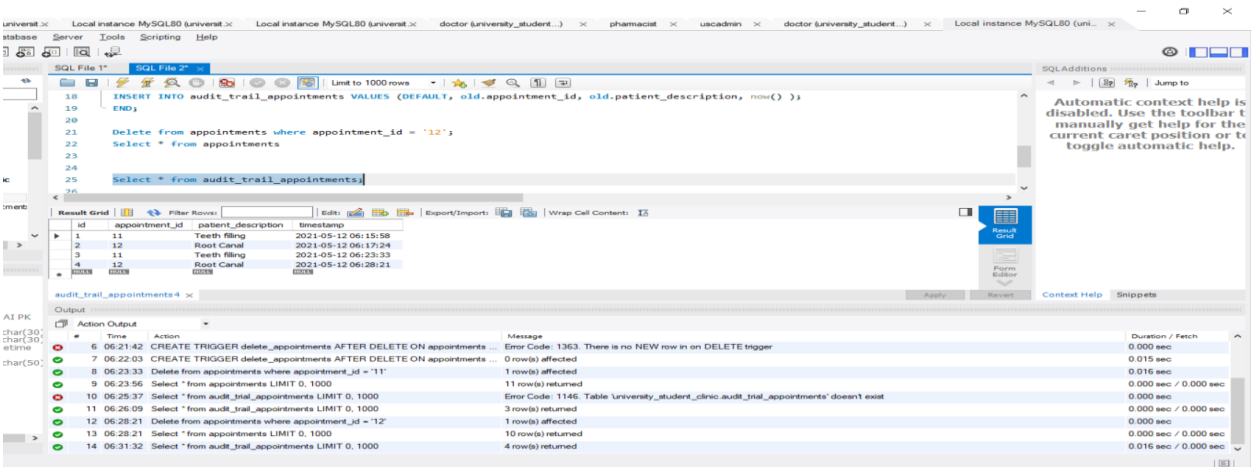
Name : Darshan Puttanna
ID: 801169012

Insert data in audit table when user perform delete action on data.

1. When particular appointments is deleted from the appointments table, that data should get updated in the audit table with timestamp.



When we ran audit_trail_appointments after deleting a row from appointments table. The deleted row is getting updated in audit_trail_appointments table, because of the delete trigger we have written.



Name : Darshan Puttanna
ID: 801169012

Triggers:

A trigger is a set of actions that are run automatically when a specified change operation is performed on a specified table. Triggers are useful for audit trail.

Types of trigger used are: After insert, before insert, after update, before update, before delete.

Requirement set 7:

1. Write trigger to add record in an audit table after medicine amount is getting updated in medicine table.

```
DELIMITER //
DROP TRIGGER IF EXISTS UPDATE_MEDICINE_AMOUNT;
DELIMITER //
CREATE TRIGGER UPDATE_MEDICINE_AMOUNT
AFTER UPDATE ON MEDICENE
FOR EACH ROW
BEGIN
INSERT INTO audit_trail_medicine VALUES (DEFAULT, NEW.medicene_name, NEW.amount, now() );
END;
```

2. Write trigger to add record in an audit table before employee phone number is updated in employee table.

```
DELIMITER //
DROP TRIGGER IF EXISTS UPDATE_doctor;
DELIMITER //
CREATE TRIGGER UPDATE_doctor
BEFORE UPDATE ON doctors
FOR EACH ROW
BEGIN
INSERT INTO audit_trail_doctor VALUES (DEFAULT, OLD.contact_number, NEW.contact_number, "doctor", now() );
END;
```

3. Write trigger to insert data in audit table after patient had booked an appointment.

Name : Darshan Puttanna
ID: 801169012

```
-- Trigger 3 and audit-insert

DELIMITER //
DROP TRIGGER IF EXISTS insert_appointments;
DELIMITER //
CREATE TRIGGER insert_appointments
AFTER INSERT ON appointments
FOR EACH ROW
BEGIN
INSERT INTO audit_trail_appointments VALUES (DEFAULT, NEW.appointment_id, NEW.patient_description, now() );
END;
```

4. Write trigger to insert data in visit_information table after checkup table is updated.

```
DELIMITER //
DROP TRIGGER IF EXISTS insert_visitinformation;
DELIMITER //
CREATE TRIGGER insert_visitinformation
AFTER INSERT ON checkup
FOR EACH ROW
BEGIN
INSERT INTO visit_information VALUES (DEFAULT, NEW.visitor_id, New.symptoms_desc, NEW.result, now() );
END;
```

5. Write trigger to insert data in audit_trail_nilling table after billing_info table is updated.

```
--Trigger 5: Billing details

DELIMITER //
DROP TRIGGER IF EXISTS insert_billing_details;
DELIMITER //
CREATE TRIGGER insert_billing_details
AFTER INSERT ON billing_info
FOR EACH ROW
BEGIN
INSERT INTO audit_trail_billing VALUES (DEFAULT, NEW.visitor_id, New.total_bill);
END;
```

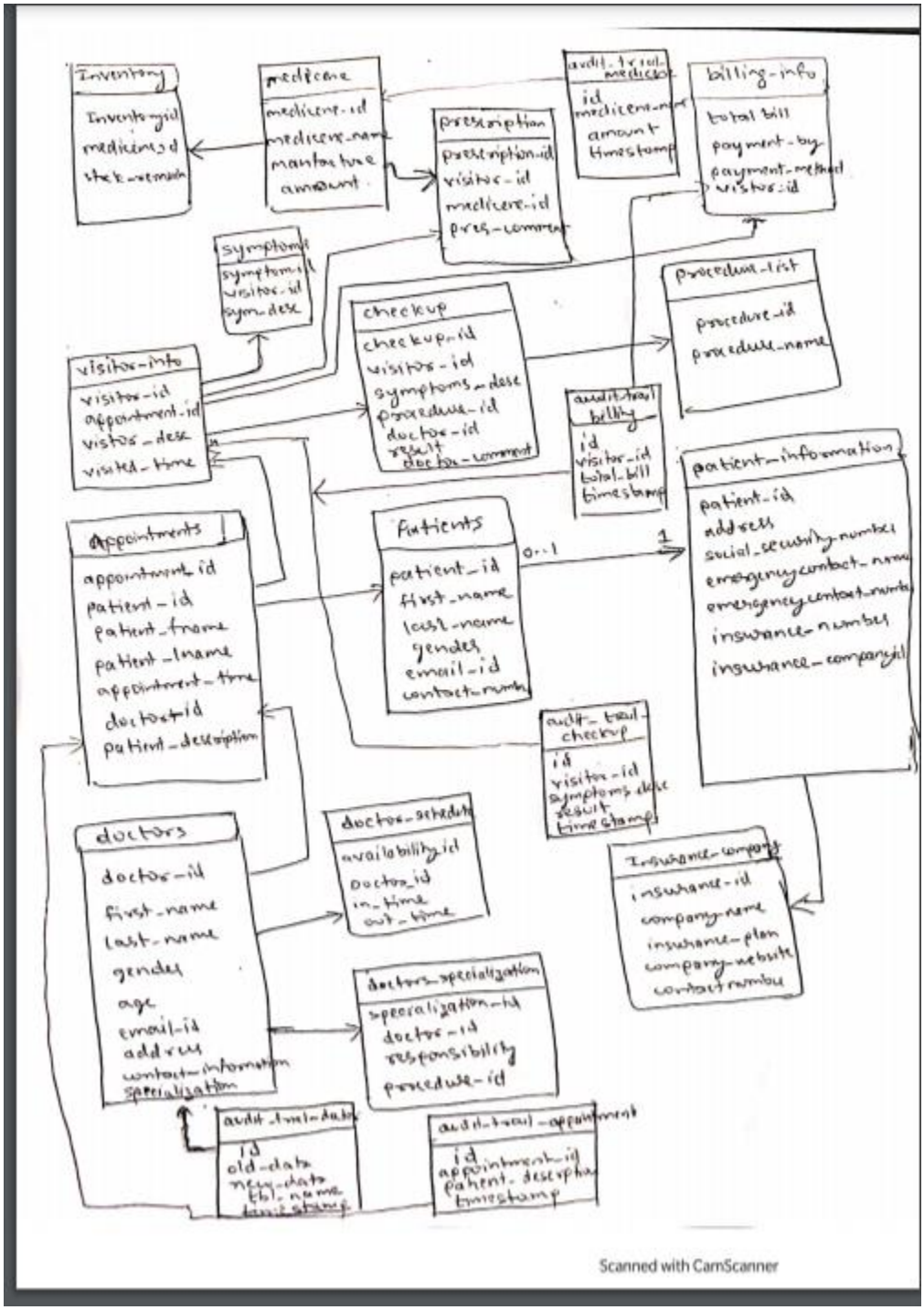
Name : Darshan Puttanna
ID: 801169012

6. Write trigger to insert data in audit table after patient had deleted an appointment.

```
-- Trigger 6 and audit-delete

DELIMITER //
DROP TRIGGER IF EXISTS delete_appointments;
DELIMITER //
CREATE TRIGGER delete_appointments
AFTER DELETE ON appointments
FOR EACH ROW
BEGIN
INSERT INTO audit_trail_appointments VALUES (DEFAULT, NEW.appointment_id, NEW.patient_description, now() );
END;
```

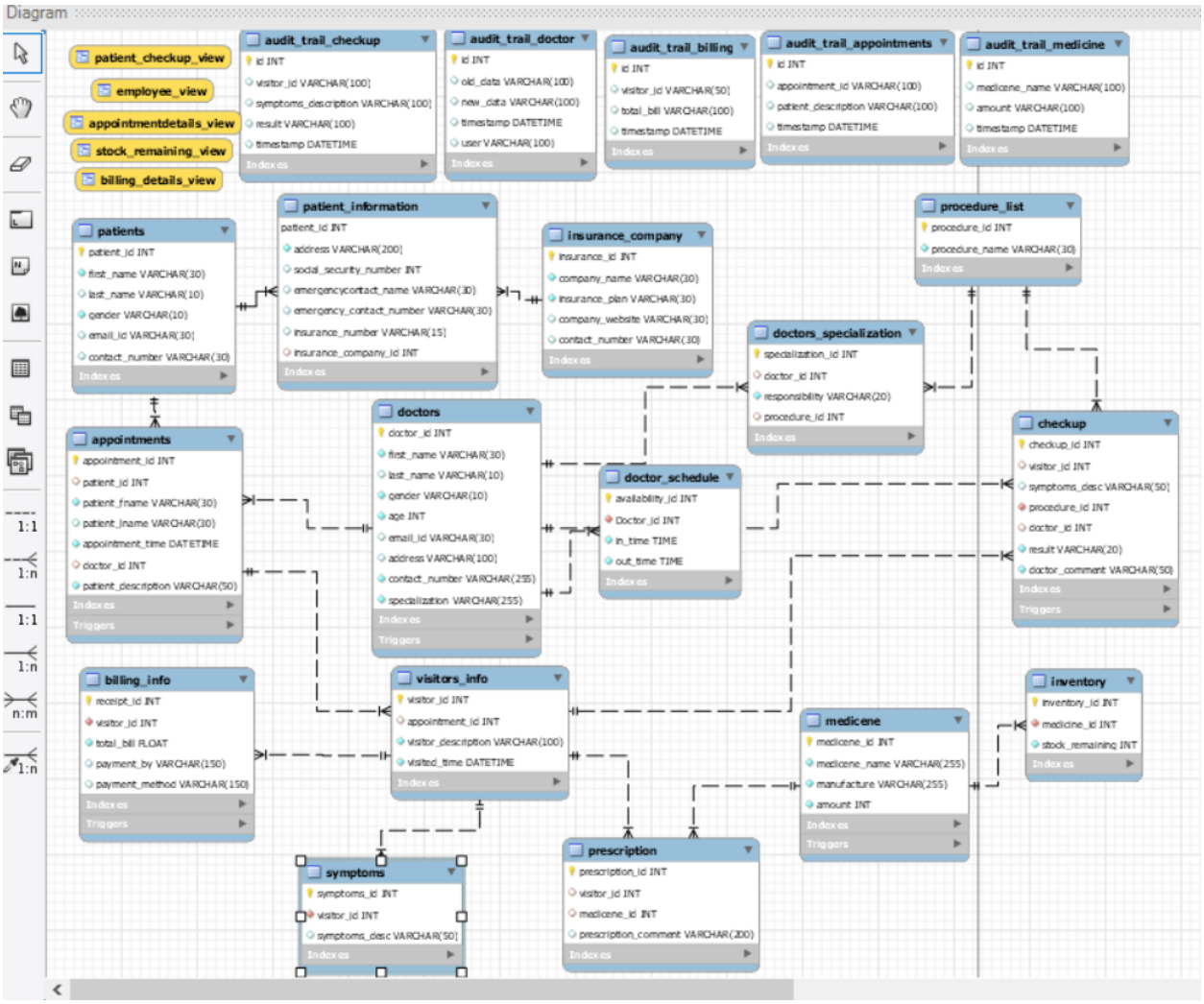
Name : Darshan Puttanna
ID: 801169012

UML Data Model



Name : Darshan Puttanna
ID: 801169012

ER Diagram



Name : Darshan Puttanna
ID: 801169012

Thank You