**Electronic Medical Record System**

(Dental Clinic)

Applied Databases – Project 2

Name: Akshara Gone

Student ID: 801136090

**Contents**

[1. Introduction: 4](#_Toc36217880)

[1.1. Purpose: 4](#_Toc36217881)

[1.2. Overview: 4](#_Toc36217882)

[2. Scope: 4](#_Toc36217883)

[3. Functional Requirements: 5](#_Toc36217884)

[4. UML Diagram: 7](#_Toc36217885)

5. ER Diagram:……………………………………………………………………………………………………………………………8

[6. Description: 9](#_Toc36217887)

[6.1. Tools Used: 9](#_Toc36217888)

[6.2. Tables Created: 9](#_Toc36217889)

[6.3. Table Contents: 9](#_Toc36217890)

[7. Application Programming Interface: 12](#_Toc39409274)

[7.1. User Authentication: 12](#_Toc39409275)

[7.2. Role-Based access: 13](#_Toc39409276)

[7.3. Stored Procedures: 15](#_Toc39409277)

[7.4. Audit Trail: 15](#_Toc39409278)

[7.5. Views: 17](#_Toc39409279)

[7.6. Indexes:…………………………………………………………………………………………..18](#_Toc39409280)

References:………………………………………………………………………………………………...19

**CHANGES MADE IN PROJECT 2**

|  |  |
| --- | --- |
| **Project 1** | 1. **Created a Database Design for EMRS – Dental Clinic** 2. **Modeled the database through ER and UML diagrams** 3. **Created relational database for the dental clinic in MySQL.** 4. **SEARCH, INSERT, UPDATE and DELETE functionality testing on the database.** |
| **Project 2**  **(changes/updates to the original database turned in for Project 1)** | 1. **Updated the UML diagram** 2. **Updated the ER diagram** 3. **Added new tables to the database** 4. **Developed API’s using stored procedures** 5. **Added users to the database** 6. **Given role based access to users** 7. **Added tables to enable audit trial** 8. **Added indexes and views** |

# Introduction:

## Purpose:

Electronic Medical Records (EMRs) or Electronic Health Records (EHRs) are considered a critical technology for improving patient care quality and reducing health care cost. An electronic database would be an ideal arrangement for maintaining the patient data. The motivation behind this project is designing and creating a database which stores a patient data for a dental clinic electronically. The Dental clinic facility sees various patients with different dental issues every day, with capturing detailed information about patient’s condition, treatment will become more evidence-based. This project ensures security robustness and reliability.

## Overview:

This project includes the design and structure of a database which includes the fundamental activities associated with the dental clinic. This is a clinical Center which admissions patients daily with various problems and arranges their visits with the respective doctors. When patient record is made, their symptoms and previous medical history are noted. In the next step, every patient is inspected by a doctor and their sickness is diagnosed by the doctor. During this stage, the patient may need to take certain tests as ordered by the Doctor. Once the test results are confirmed, later the doctor gives a prescription with the drugs needed to be taken for the diagnosis. This Medical Record System predominantly focuses on the patients, their visits to the clinic, diagnosis of the illness, tests, prescription and billing. This Medical Record system is kept up electronically through the Database which contains the information of the patients split across various tables.

# Scope:

The primary goal of this project is to store the patient related records electronically through a database which includes the assortment of all the patient records, details of their visits and the diagnostics data. Additionally, the aim is to establish a connection between the tables and split the information in various tables. Doing this, the information recovery can be made simpler and pertinent information can be obtained whenever necessary.

# Functional Requirements:

There are various entities involved in this system which are listed as below:

**Doctor**

Doctor is a health care professional who examines the patient and diagnosis the illness of the patients. Here all doctors are dentists, and they treat various dental issues accordingly. To diagnose the patient precisely doctors, order some tests if necessary.

**Patients**

Every patient record should store data like Name, Insurance provider etc. A patient can have numerous visits for various sickness. Since the patient can visit the clinic for various necessities they can be assigned under different doctors for each visit. In my system patient is the central entity, and it will connect every other entity.

**Patient Contact Details:**

Patient Contact Details is mainly used to store the patient demographic information. This information can be stored in the Patients table, but to ensure tables are in BCNF separated this into another table.

**Insurance Company**

Insurance data is very important for storing patients record. This data includes details like InsuranceCompanyID, ProviderName, ProviderLocation, ProviderContact, ProviderEmail.

**Visits**

Visit table includes the movement when the patient visits the clinic and leaves the clinic. Each visit record will hold the patient data, exam-room assigned, diagnosis, prescription and billing information. A visit record is an important entity and maintains relation with different entities in the electronic medical records system.

**Exam rooms**

The examination room is normally the entry point for a patient into the hospital. Visual examination plays a major role in deciding the right treatment. Each time a patient visits the clinic they would be assigned an examination room. There are different types of exam rooms.

**Diagnosis:**

The doctor will give the diagnosis report of the patient depending on the patients dental issue. The diagnosis information is unique to each visit.

**Prescription:**

After concluding the diagnosis, the doctor recommends necessary medicinal drugs for the patient to recover.

**Patient\_Bills:**

Each visit of the patient is charged independently. The billing sheet incorporates total amount charged, insurance amount and the balance due.

**Medicine\_Supplier\_Company:**

The medicines are supplied by various companies. One company can supply many medicines.

**Medicines:**

This table keeps track of all the medicines in the clinic.

**Users (added as part of the Project 2)**

The application user details like username and password are stored in this table, each user is associated with a role like doctor, clerk and a nurse. The operation on DB are given based on user roles.

# UML Diagram:

Diagram

Description automatically generated

The model consists of associations like aggregation and composition.

I have ensured that the tables are in BCNF.

**changes/updates to original database from Project 1:**

7 new tables are added to the dental\_clinic database as part of project 2,

Users – table stored the application user data like username, password, and role

billing\_audit\_table – keeps track of the history of data in billing table

diagnosis\_audit\_table – keeps track of the history of data in diagnosis table

doctors\_audit\_table – keeps track of the history of data in doctors table

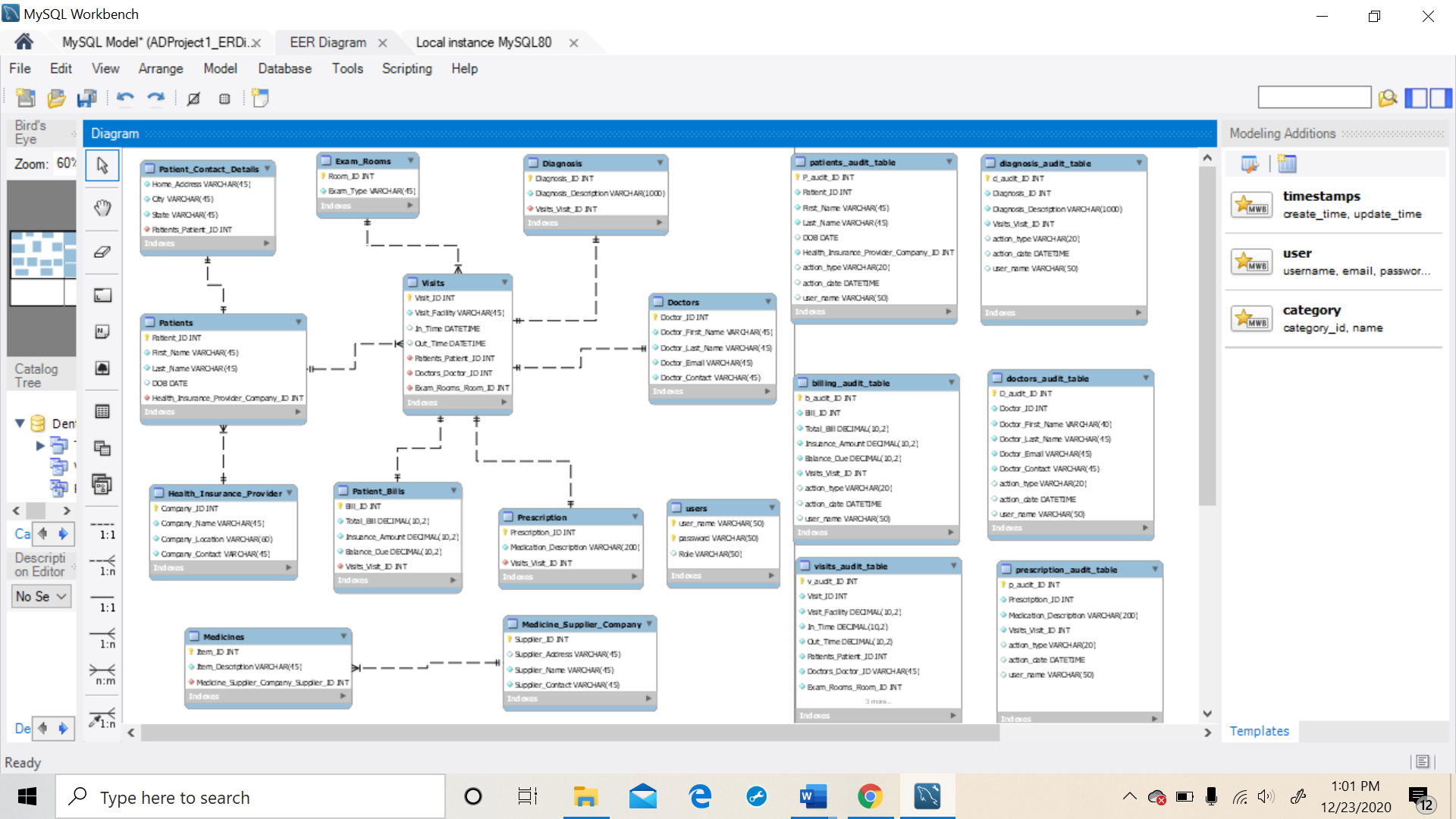
patients\_audit\_table – keeps track of the history of data in patients table

visits\_audit\_table – keeps track of the history of data in visits table

prescription\_audit\_table – keeps track of the history of data in prescription table

# ER Diagram:

An ER model is a representation of the entities, or objects of the database including the tables, views and stored procedures. It provides a visual representation of the relationships between the tables often used to define those relationships.



**changes/updates to original database from Project 1:**

7 new tables are added to the electronic medical record system database as part of project 2,

Users

billing\_audit\_table

diagnosis\_audit\_table

doctors\_audit\_table

patients\_audit\_table

visits\_audit\_table

prescription\_audit\_table

# Description:

To store the data efficiently and to recover it effectively, I have designed and structured a database with 11 tables which have relationship among themselves so that instead of putting away all data in a single table, this data can be separated into multiple relevant tables. At any point, if data is required, user can compose a query and get the significant information.

## Tools Used:

* MySQL WorkBench 8.0

## Tables Created:

1. Doctors
2. Patients
3. Patient\_ Contact\_Details
4. Visits
5. Exam\_Rooms
6. Diagnosis
7. Health\_Insurance\_Company
8. Patient\_Bills
9. Prescription
10. Medicine\_Supplier\_Company
11. Medicines

**Tables are added as part of project 2**

1. Users
2. Billing\_audit\_table
3. Diagnosis\_audit\_table
4. Doctors\_audit\_table
5. Patients\_audit\_table
6. Visit\_audit\_table
7. Prescription\_audit\_table

## Table Contents:

* 1. **Doctors**:

The Doctors table contains the following attributes

1. DoctorID (Primary Key)
2. Doctor\_First\_Name
3. Doctor\_LastName
4. Doctor\_Phone
5. Doctor\_Email
   1. **Patients:**

The “Patients” table contains the following attributes

* + 1. PatientID (Primary Key)
    2. Patient\_First\_Name
    3. Patient\_Last\_Name
    4. Patient\_DOB
    5. Insurance\_Company\_ID(Foreign key)
  1. **Patient\_Contact\_Details:**

The “Contact\_Details” table contains the following attributes

* + 1. Patient\_ID (Primary Key)
    2. Home\_Address
    3. City
    4. state
  1. **Visits:**

The “Visits” table contains the following attributes

* + 1. Visit\_ID (Primary Key)
    2. Patient\_ID (Foreign key)
    3. Visit\_Facility
    4. In\_time
    5. Out\_time
    6. Doctor\_ID (Foreign key)
    7. Exam\_room\_ID (Foreign key)
  1. **Exam\_Rooms:**

The “Exam\_Rooms” table contains the following attributes

* + 1. Exam\_room\_ID (Primary Key)
    2. Exam\_room\_type
  1. **Diagnosis:**

The “Diagnosis” table contains the following attributes

* + 1. Diagnosis\_ID (Primary Key)
    2. Diagnosis\_Description
    3. Visit\_ID (Foreign Key)
  1. **Health\_Insurance\_Company:**

The “InsuranceCompany” table contains the following attributes

* + 1. Insurance\_Company\_ID (Primary Key)
    2. Company\_Name
    3. Company\_Location
    4. Company\_Contact
  1. **Patient\_Bills:**

The “Billing” table contains the following attributes

* + 1. Bill\_ID (Primary Key)
    2. Visit\_ID (Foreign Key)
    3. Total\_Bill
    4. Insurance\_Amount
    5. Balance\_Due
  1. **Prescription:**

The “Prescription” table contains the following attributes

* + 1. Prescription\_ID (Primary Key)
    2. Medication\_Description
    3. Visit\_ID (Foreign Key)
  1. **Medicine\_Supplier\_Company:**

The “Suppliers” table contains the following attributes

* + 1. Supplier\_ID (Primary Key)
    2. Supplier\_Address
    3. Supplier\_Name
    4. Supplier\_Contact
  1. **Medicines:**

The “Supplies” table contains the following attributes

* + 1. Item\_ID (Primary Key)
    2. Item\_description
    3. Supplier\_Id (Foreign Key)
  1. **Users: (added as part of the Project 2)**

The “Users” table contains the following attributes

* + 1. UserName (Primary Key)
    2. Password (Primary Key)
    3. Role

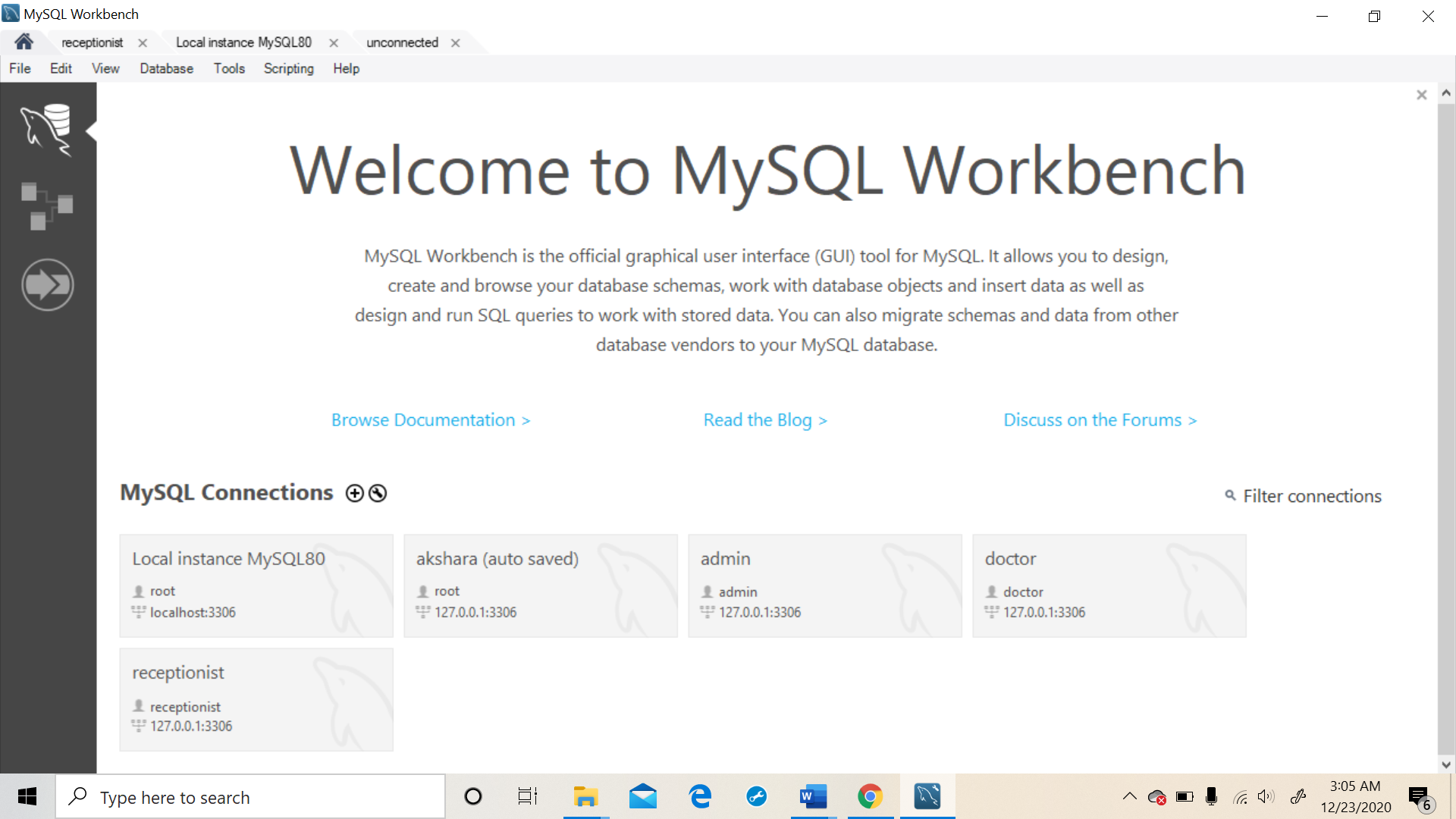
# Application Programming Interface: **(This includes Stored Procedures, Triggers which are added as part of Project 2)**

The Application programming is created using stored procedures. EMR System users can work with the database without having to write any SQL statements. Stored Procedures, triggers and views allows the developers to work with the database without having to write SQL statements. The stored procedures are called by the developers.

## User Authentication:

User authentication is added so that only authorized users can access the database. The authorized users must provide password to login to the system.

|  |  |
| --- | --- |
| Authorized Users |  |
| User name | Password |
| admin | admin |
| doctor | doctor |
| receptionist | receptionist |



## Stored Procedures:

Stored Procedure or procedure is a database object that contains a block of procedural SQL code. We can use stored procedures to modify the data that is stored with in a database. Stored procedure can be used to execute a SELECT, INSERT, UPDATE or DELETE statement.

List of stored procedures created:

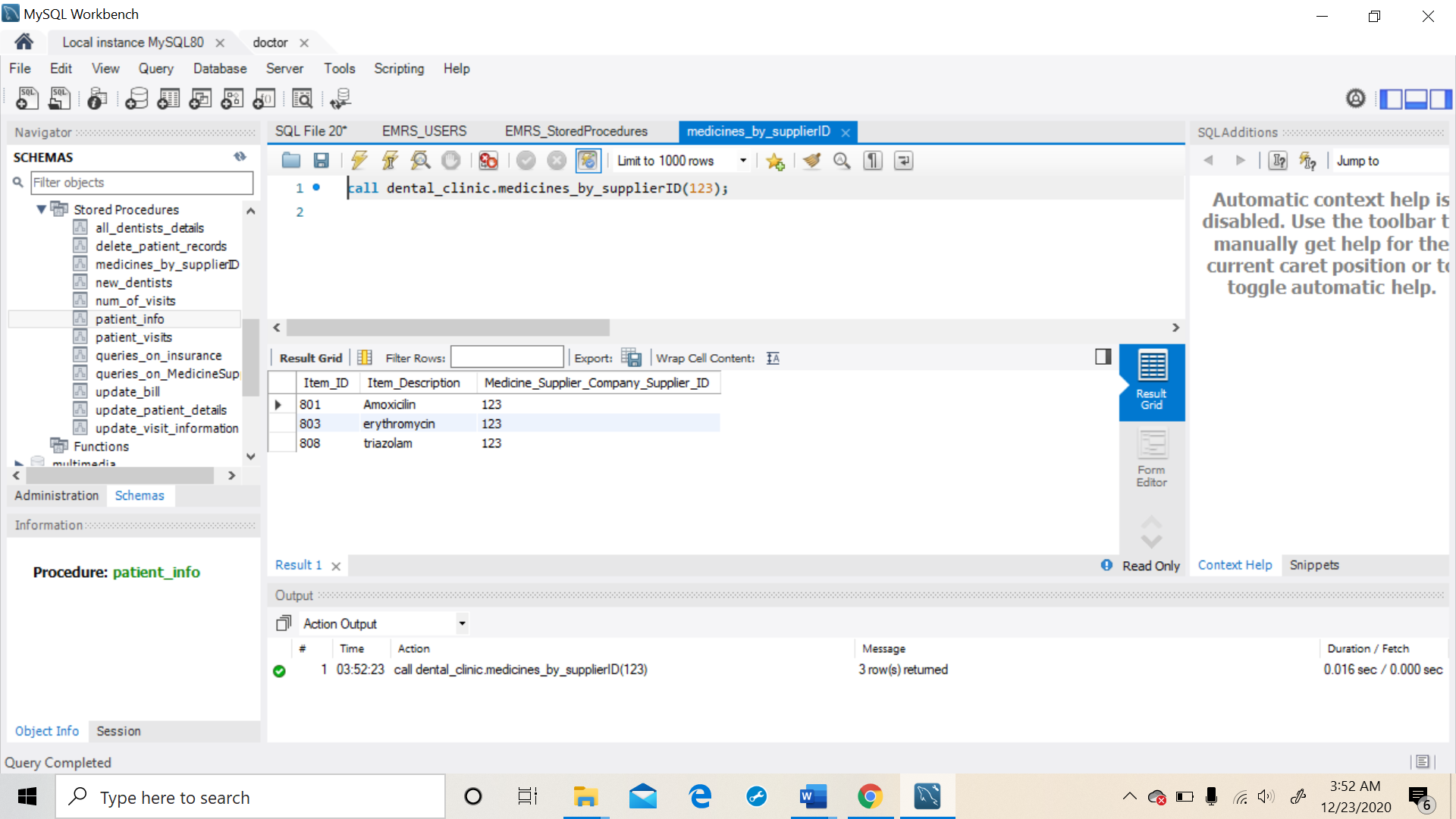
|  |  |  |
| --- | --- | --- |
|  | Stored procedures used in this Project | |
| **S.NO** | **Procedure\_Name | Action** | |
| 1 | medicines\_by\_supplierID | display all the medicines produced by a given medicine\_supplier\_id as input by the user |
| 2 | patient\_visits | display all the visits made by a patient, given his first and last name as input by the user |
| 3 | all\_dentists\_details | display details of all the dentists in the hospital |
| 4 | update\_bill | used to update a patients bill |
| 5 | num\_of\_visits | display number of visits made by all the patients |
| 6 | update\_patient\_details | update a patient\_contact\_details when their address is changed. |
| 7 | new\_dentists | insert a new dentist record whenever a new dentist joins the hospital. |
| 8 | queries\_on\_insurance | perform Update, Insert or Delete operations on the health\_insurance\_provider table |
| 9 | delete\_patient\_records | delete a patients record from the hospital database |
| 10 | update\_visit\_information | update the reason for hospital visit of a patient information in the visits table |
| 11 | queries\_on\_MedicineSuppliers | perform Update, Insert or Delete operations on the medical\_supplier\_company table |
| 12 | patient\_info | view all the patients complete Information by joining the patients and patient\_contact\_details information |

**TestCase: The user need not write the SQL script to check what medicies are supplied by a specific supplier company. The user should just Run the stored procedure ‘medicines\_by\_supplierID’ and enter the Supplier\_ID in the column as shown below.**

A screenshot of a computer

Description automatically generated

**After entering the Supplier\_ID, execute the procedure and the output is displayed as shown below.**



**In this way, users can perform operation on the database using Stored Procedures without actually writing the SQL scripts.**

## Role-Based access:

|  |  |  |
| --- | --- | --- |
| Admin | Doctor | Receptionist |
| Has **access to all the tables** in the database.  Can perform **SELECT, INSERT, UPDATE and DELETE** on any table in the database. | **INSERT, UPDATE, SELECT** access on the below listed tables.  Patients  Patients\_contact\_details  Diagnosis  Prescription  Visits  **No DELETE** access on any table. | **INSERT, UPDATE and DELETE** access on the below listed tables  Patients  Patients\_contact\_details  Visits  Patient\_Bills |

## Audit Trail:

Audit trail is used to maintain a record of the changes that have been made to a database. It is important to have the list of past changes to the records and this can be achieved with the help of triggers. Triggers can be characterized as a database object that is related with a table. It will be activated when a defined action is executed for the table. By utilizing triggers, we do not have to wait for the scheduled events to run because the triggers are invoked automatically before or after a change is made to the data in a table.

I have 6 audit tables created in my project as listed below

|  |  |
| --- | --- |
| Audit Tables | |
| billing\_audit\_table | Tracks the actions:   * actions performed on the table * time at which the respective action is performed * the user who performed the action |
| diagnosis\_audit\_table |
| doctors\_audit\_table |
| patients\_audit\_table |
| visits\_audit\_table |
| prescription\_audit\_table |

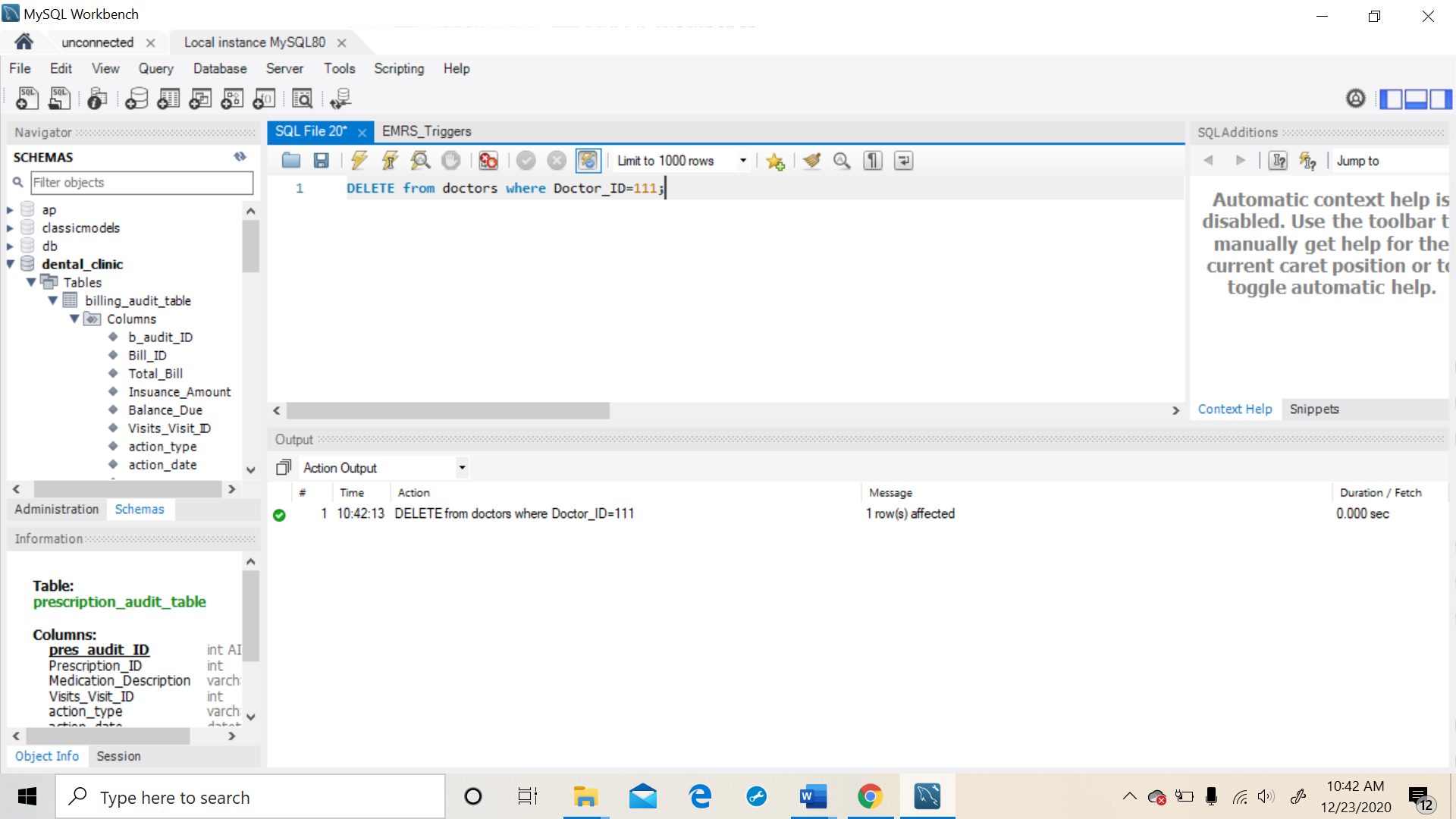
Audit tables are not created for few tables because the data in those tables is fixed and either no changes would be made to them or a very few changes would be made over time.

For example, the exam\_rooms in a hospital are fixed and would remain unchanged. So, audit trail is not maintained for some of these tables.

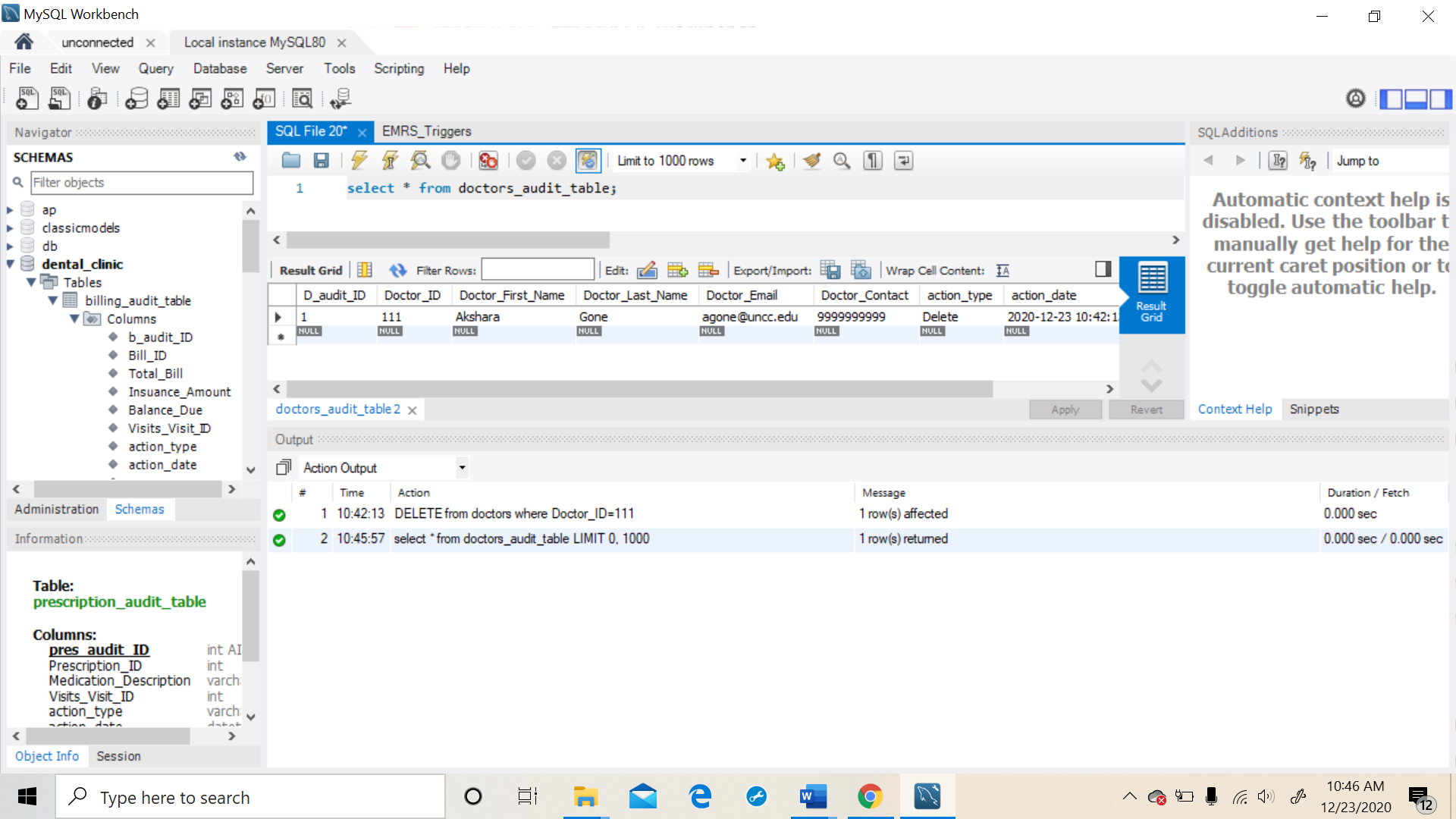
**Doctors Audit table:**

Delete operation made on doctors table is captured in doctor\_audit\_table. The action\_type = DELETE, action\_date = the time at which delete is performed, and the user\_name shows the user who performed the delete.

**Example: Delete a record from doctors table whose ID = 111**



**Resultant values are recorded in doctor\_audit\_table as shown below**



Similarly UPDATE and DELETE operations executed on doctors table are also captured in the doctor\_audit\_table.

In the same way, any operations performed on these tables are recorded in their respective audit tables with the help of triggers.

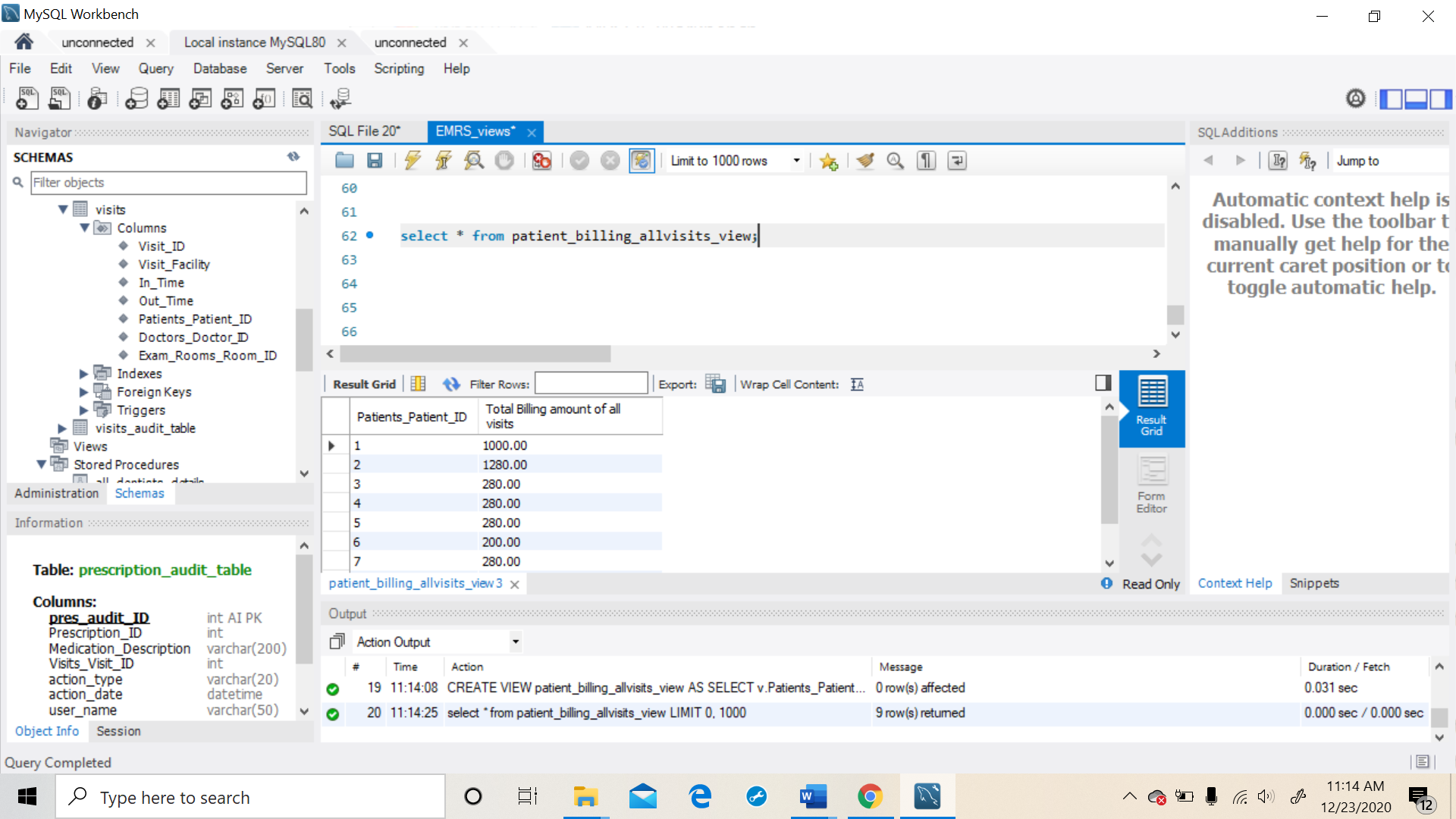
## Views:

Views are pre-written queries that when invoked produce a result set. View acts as a virtual table. Views can be helpful to hide table columns from users by granting them access to the view and not to the table. This helps enhance database security and integrity. Reports are created in order to advance the search option and fetch the data faster.

List of views created in this project are:

|  |  |
| --- | --- |
| Views | |
| patient\_bill\_view | patient bill details for each visit |
| patient\_billing\_allvisits\_view | patient ID and their total billing amount of all visits to clinic |
| patient\_diagnosis\_view | shows a view of patient visits, the doctors visited, and diagnosis given |
| patient\_prescription\_view | view that displays patient name, ID, doctor name and medication details in each visit to the hospital |
| patient\_visits\_view | patients and their visits |

**Below is the view that displays the bill made by all patients by combining all visits using the patient\_billing\_allvisits\_view**



There are other views implemented in this project, from which the users can view the consolidated details.

## Indexes:

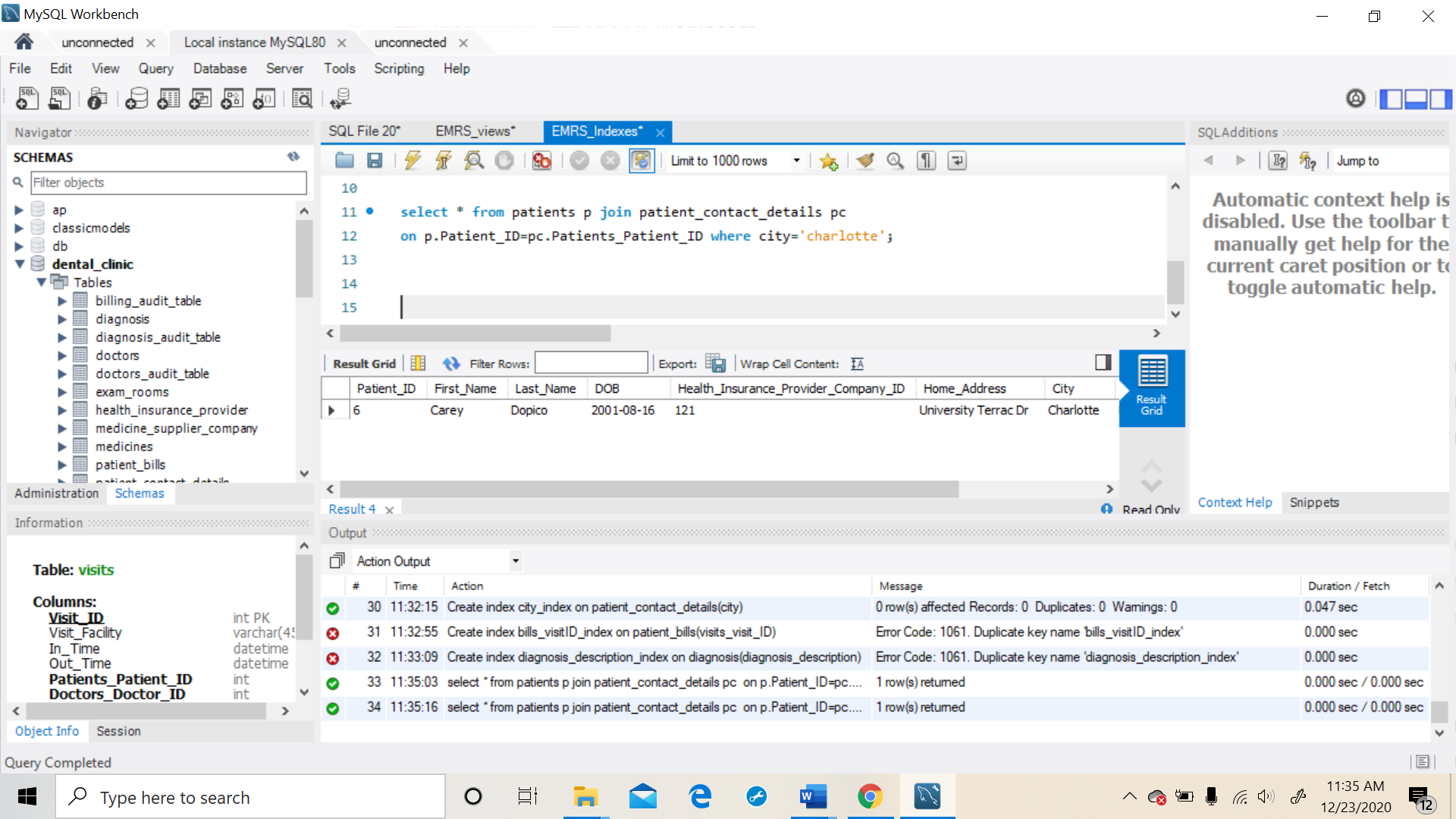
Indexes are used to find rows with specific column values quickly. MySQL can quickly determine the position to seek to in the middle of the data file without having to look at all the data. This is much faster than reading every row sequentially. Indexes are used to eliminate rows from consideration and make the search operation faster.

In the database, I have created indexes for most of the tables wherever I wanted faster outputs from the database for certain search criteria.

|  |  |
| --- | --- |
| Indexes | |
| Table name | Index created on |
| diagnosis | diagnosis\_description |
| doctors | Doctor\_Last\_Name |
| patients | Last\_Name |
| patient\_contact\_details | city |
| prescription | Visits\_Visit\_ID |
| Patient\_bills | Visits\_Visit\_ID |
| visits | In\_Time |

**Test Case 6: patient\_contact\_details have an index on city**

We can see that the search was faster, and we were able to retrieve the results faster because the indexes have been implemented.



# References:

* Murach’s MySQL – 2nd Edition