Lewis University  
CPSC 50900: Database Systems  
Term Project

DENTAL\_HOSPITAL\_MANAGEMENT

Venkata Gopi Mandalapu , venkatagopimanadala@lewisu.edu

Kolan Harilok Reddy ,HarilokReddykolan@lewisu.edu

Srikanth palle , Srikanthpalle@lewisu.edu

https://github.com/harilok512/DENTAL\_HOSPITAL\_MANAGEMENT\_DBMS

Table of Contents

[Initial Proposal 2](#_Toc85814569)

[Data Sources 2](#_Toc85814570)

[Data Storage Alternatives 2](#_Toc85814571)

[Relational Database Design Process 3](#_Toc85814572)

[Relational Database Design 3](#_Toc85814573)

[Data Definition Language (DDL) Scripts 3](#_Toc85814574)

[Data Manipulation Language Scripts 4](#_Toc85814575)

[Indexes 5](#_Toc85814576)

[Views 5](#_Toc85814577)

[Triggers 5](#_Toc85814578)

[Transactions 5](#_Toc85814579)

[Database Security 6](#_Toc85814580)

[Locking and Concurrent Access 6](#_Toc85814581)

[Backing Up Your Database 6](#_Toc85814582)

[Python Programming 7](#_Toc85814583)

[PHP Programming 7](#_Toc85814584)

[Suggested Future Work 8](#_Toc85814585)

[Activity Log 8](#_Toc85814586)

# Initial Proposal

*Description: You will describe the data you aim to store. What data will be storing? Why are you interested in this data? Why is it important? Where will the data come from? Who will use this data? What kind of application do you plan to build with it?*

*Rubric: Your response to each of these six questions will be graded out of 3 points.*

* *3 points: clear, complete descriptions that convey the importance and meaning of your data*
* *2 points: mostly clear descriptions, although some additional data would have helped in some sections*
* *1 point: necessary details are lacking in many of your responses.*

*You will also earn 2 additional points for coming up with a descriptive title for your project.*

*Total points possible: 20*

Importance of data: The main objective of this database is to build dental hospital management system. It is important to manage the details of appointments ,patients ,doctors ,bills etc. The purpose of this application is to reduce the manual work for managing the records of patients ,doctors ,bills ,etc

The data records has been collected from the dental hospital located in our locality

# Data Sources

*Description:* *Gather your data in text files. The text files may be csv, tab-delimited, xml, json, or some other custom format. Not all the files need be of the same type. Identify what each file contains by indicating where it came from, explaining in detail how it structured, and describing how you will reorganize the data into a relational database. Post your data files to your GitHub repository, and provide samples of the data in your Word doc.*

*Rubric: Your work will be graded as follows:*

* *5 points: you gathered multiple data files that contain the data that will populate your databases. If you do not use multiple data files, you will not receive credit.*
* *5 points: you described the contents of the data files in detail, including referencing their origin and explaining how they were structured.*
* *3 points: you identify which fields you plan to include in your database, including their data types and any constraints you expect to impose on the data or steps you'll have to take to clean up the data.*
* *2 points: you post the data files to your GitHub account and make it possible for me to see them.*

*Total points possible: 15*

ENTER YOUR DATA SOURCES DESCRIPTION HERE

• Details: The dataset is the collective information the details of appointments ,patients ,doctors ,bills etc . The data files provided in the github repository https://github.com/harilok512/DENTAL\_HOSPITAL\_MANAGEMENT\_DBMS

• Origin: The data used was collected from the local dental clinic and in was in the form of jornal where the whole data was written in rows and coloums . The data found was totally in the written form

•we primarily filtered the data by separated them in further tables by their respected fields such patients , doctors , appointments . Then the was transferd into csv ,Xml ,Json format which can be used to work across various platforms.

• The data was cleaned by fields and we have put on new fields like appoinments and transactions which was not present previously so that the whole database will be of proper structure.

# Data Storage Alternatives

*Description: We will study alternatives to storing data in a relational database. Some of the alternatives come from several decades ago, including the hierarchical and network models. Some are newer options, such as NoSQL databases that use JSON or some other encoding. Describe in detail how to store the data using two alternatives to relational databases. Be sure to describe how you would implement the alternatives and the advantages and disadvantages of each.*

*Rubric: Your work will be graded as follows*

* *5 points for clearly describing how your data could be stored using one alternative to relational databases and what the advantages and disadvantages of that approach would be.*
* *5 points for clearly describing how your data could be stored using another alternative to relational databases and what the advantages and disadvantages of that approach would be.*

*Total points possible: 10*

If I must choose an alternative to store data in a relational database, I would go with NoSQL databaseparticularly MongoDB database. This database contains a storage unit called ‘Collections’ whichcorresponds to ‘Table’ in relational database. This collection contains the data in the form of documents injson format. Each document corresponds to a single record in a table.Advantages:

• Absence of complex joins in this database.

• It doesn’t require schema to be built before storing the data.

• Scaling is easy.Disadvantages:

• If no correct indexing is done for the data, the database can operate at shockingly slowspeed.

• Absence of Transactions regarding data.

My second alternative would be graph database. This database stores not only data but also therelationships between the data, making queries about relationships a lot faster. Since data are physicallylinked in the database, query execution is basically traversing the graph.Advantages:

• Excellent performance during querying huge datasets.

• It is schema less which means it doesn’t require in which format data should be stored.

• Best choice for data that contains a rich relationship.Disadvantages:

• Doesn’t have a uniform query language.

• Simply provides faster data retrieval for interlinked data but not in creating betterrelationships between them.

• Not optimized for queries covering almost the entire database.

# Relational Database Design Process

*Description: Consider the list of fields you identified in part c. Identify functional dependencies that exist among them. For each functional dependency, identify the determinants and the fields they determine. This becomes the basis for identifying your entity sets, which then become your tables. Give each entity set or table you identify in this way a unique and clear name, making sure that the names you use are singular nouns. Then list the relationships that exist among the various entity sets. For each relationship, identify its connectivity (one-to-one, one-to-many, many-to-many) and participation (optional or mandatory). Finally, make sure that none of the attributes you've assigned to each entity set are multi-valued. If they are, take the steps needed to break them down.*

*Rubric: Your work will be graded as follows:*

* *8 points for identifying all the functional dependencies, including determinants and the columns whose values they determine.*
* *2 points for naming the entity sets that make up your data with clear, easy-to-understand names.*
* *6 points for identify the relationships among the entity sets and identifying connectivity and participation for each.*
* *2 points for breaking down multi-valued attributes.*

*Total points possible: 18*

ENTER YOUR RELATIONAL DATABASE DESIGN DESCRIPTION HERE. INCLUDE SOURCE CODE AND SCREEN SHOTS.

Functional dependencies are Doctor-id->doctor title, Appointment-id->appointment, patient id->patient name, treatment id-> treatmentname, transaction id->bill and the determinants include patient id, treatment id, doctor id, transaction id, Appointment id.

Entity sets are doctor, Patient, Bills, Appointment, Transactions and Treatment.

Patient - Appointment: one-to-many

Doctor - Appointment: one-to-many

Appointment-Transaction: one-to-one

Transactions - Bills: one-to-one

Treatment - transactions: one-to-one

# Relational Database Design

*Description: This is where you will complete your database design. For each of the entity sets you identified in the preceding section, analyze them to make sure they pass 2nd, 3rd, 4th, and Boyce-Codd Normal Form. If they do not, introduce additional entity sets or key changes to make sure that they do. Then, add foreign keys to connect entity sets that are related. For many-to-many relationships, introduce bridge entity sets to convert them into two one-to-many relationships. Also, consider whether you should introduce surrogate keys to create a more efficient primary key for some of your entity sets. Finally, diagram your design in Vertabello. Make sure your ER diagram correctly shows all entity sets, their primary and foreign keys, the data types for each attribute, and the connectivity and participation characteristics of each entity set. Your final Vertabello design should be something you could actually implement in a relational database management system.*

*Rubric: Your work will be graded as follows:*

* *4 points for the normalization analysis of your entity sets.*
* *3 points for introducing bridge entity sets.*
* *3 points for choosing foreign keys and perhaps more efficient surrogate keys*
* *10 points for correctly depicting your physical database model in Vertabello*

*Total points possible: 20*

ENTER YOUR RELATIONAL DATABASE DESIGN HERE

Here the bridge set is transactions which includes transaction id, bill amount, patient id, treatment id and doctor id.

Foreign keys are patient id, treatment id and doctor id.

Diagram

Description automatically generated

# Data Definition Language (DDL) Scripts

*Description: Use Vertabello to generate a script of SQL commands that build the database and its table structures. Write scripts or build Excel spreadsheets that take your data files and generate scripts of SQL insert statements from them. Use the MySQL source command to run the various scripts needed to build and populate the database in MySQL. Include the source code and / or Excel spreadsheets you use to manipulate and populate the data. Make sure all your tables have at least three records in them and that you've linked the tables through their foreign keys.*

*Rubric: Your work will be grades as follows:*

* *Database and table creation statements from Vertabello saved as an sql script file: 3 points*
* *Scripts you write or Excel spreadsheets you create to generate SQL commands for populating the tables, uploaded to GitHub: 8 points*
* *Descriptions of the scripts and Excel spreadsheets you wrote along with code excerpts included in the Word document: 5 points*
* *Screenshots of your successful attempts to use the MySQL source command to populate each table with at least three records: 4 points*

*Total points possible: 20*

ENTER YOUR DDL WORK HERE

The documents include are dental management create.sql and dental\_insert.sql.These documents include the creation and insertion statements

-- tables

-- Table: Appointment

CREATE TABLE Appointment (

Appointment\_ID int NOT NULL,

Appointment\_Date int NOT NULL,

patient\_patient\_id varchar(50) NOT NULL,

Doctor\_Doctor\_Id varchar(50) NOT NULL,

Transactions\_Transaction\_id varchar(50) NOT NULL,

CONSTRAINT Appointment\_pk PRIMARY KEY (Appointment\_ID)

);

-- Table: Bills

CREATE TABLE Bills (

Bill\_id varchar(50) NOT NULL,

Bill\_amount int NOT NULL,

Transactions\_Transaction\_id varchar(50) NOT NULL,

CONSTRAINT Bills\_pk PRIMARY KEY (Bill\_id)

);

-- Table: Doctor

CREATE TABLE Doctor (

Doctor\_Id varchar(50) NOT NULL,

Doctor\_name varchar(50) NOT NULL,

doctor\_number int NOT NULL,

Specialization varchar(50) NOT NULL,

CONSTRAINT Doctor\_pk PRIMARY KEY (Doctor\_Id)

);

-- Table: Transactions

CREATE TABLE Transactions (

Transaction\_id varchar(50) NOT NULL,

Bill\_id varchar(50) NOT NULL,

Treatement\_treatement\_id varchar(50) NOT NULL,

CONSTRAINT Transactions\_pk PRIMARY KEY (Transaction\_id)

);

-- Table: Treatement

CREATE TABLE Treatement (

treatement\_id varchar(50) NOT NULL,

treatement\_name varchar(50) NOT NULL,

CONSTRAINT Treatement\_pk PRIMARY KEY (treatement\_id)

);

-- Table: patient

CREATE TABLE patient (

patient\_id varchar(50) NOT NULL,

patient\_name varchar(25) NOT NULL,

phn\_number int NOT NULL,

age int NOT NULL,

Address varchar(30) NOT NULL,

CONSTRAINT patient\_pk PRIMARY KEY (patient\_id)

);

-- foreign keys

-- Reference: Appointment\_Doctor (table: Appointment)

ALTER TABLE Appointment ADD CONSTRAINT Appointment\_Doctor FOREIGN KEY Appointment\_Doctor (Doctor\_Doctor\_Id)

REFERENCES Doctor (Doctor\_Id);

-- Reference: Appointment\_Transactions (table: Appointment)

ALTER TABLE Appointment ADD CONSTRAINT Appointment\_Transactions FOREIGN KEY Appointment\_Transactions (Transactions\_Transaction\_id)

REFERENCES Transactions (Transaction\_id);

-- Reference: Appointment\_patient (table: Appointment)

ALTER TABLE Appointment ADD CONSTRAINT Appointment\_patient FOREIGN KEY Appointment\_patient (patient\_patient\_id)

REFERENCES patient (patient\_id);

-- Reference: Bills\_Transactions (table: Bills)

ALTER TABLE Bills ADD CONSTRAINT Bills\_Transactions FOREIGN KEY Bills\_Transactions (Transactions\_Transaction\_id)

REFERENCES Transactions (Transaction\_id);

-- Reference: Transactions\_Treatement (table: Transactions)

ALTER TABLE Transactions ADD CONSTRAINT Transactions\_Treatement FOREIGN KEY Transactions\_Treatement (Treatement\_treatement\_id)

REFERENCES Treatement (treatement\_id);

-- End of file.

insert into Doctor values('DD543','Prathusha',983459832,'Dental surgey');

insert into Doctor values('DD098','Dexter',68236592,'Clinical dentistry');

insert into Doctor values('DD102','Micheal',23473956,'Endodontics');

insert into Appointment values(101, '21/02/2021', 'DP003', 'DD543', 'TS0023');

insert into Appointment values(102, '05/10/2021', 'DP004', 'DD098', 'TS0024');

insert into Appointment values(103, '15/09/2020', 'DP004', 'DD098', 'TS0109');

insert into Appointment values(104, '04/04/2019', 'DP001', 'DD102', 'TS0025');

insert into patient values('DP001','Vikram',84567643,30,'New jersey');

insert into patient values('DP002','Stephen',83251890,27,'New jersey');

insert into patient values('DP003','Ronald',546439335,22,'Newark');

insert into patient values('DP004','Suzy',437999217,32,'New jersey');

insert into patient values('DP005','Rose',549278460,24,'Trenton');

insert into Bills values('DWK3389P',75,'TS0023');

insert into Bills values('DWK2679Y',200,'TS0024');

insert into Bills values('DWK8729U',170,'7367894VCY');

insert into Bills values('DWK0991P',250,'2141718DHC');

insert into Bills values('DWK7233T',100,'TS0025');

insert into Bills values('DWK6778P',100,'3968892YET');

insert into Bills values('DWK0029E',85,'TS0109');

insert into Transactions values('TS0023','DWK3389P', 'TT1001');

insert into Transactions values('TS0024','DWK2679Y', 'TT1002');

insert into Transactions values('TS0025','DWK7233T', 'TT1004');

insert into Transactions values('TS0109','DWK0029E', 'TT1005');

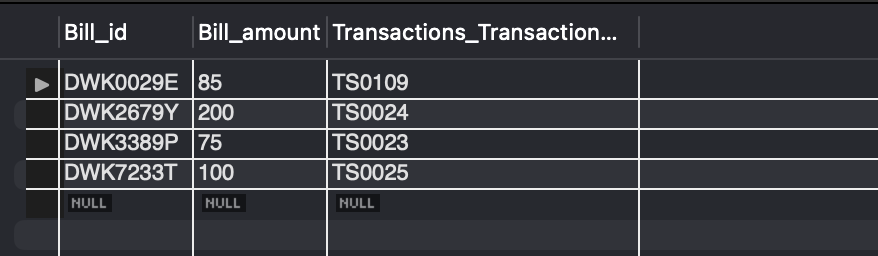
insert into Treatement values('TT1001','molar removal');

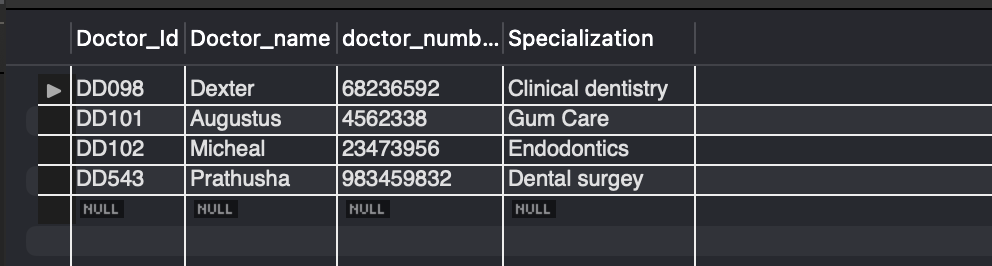
insert into Treatement values('TT1002','Normal Checkup');

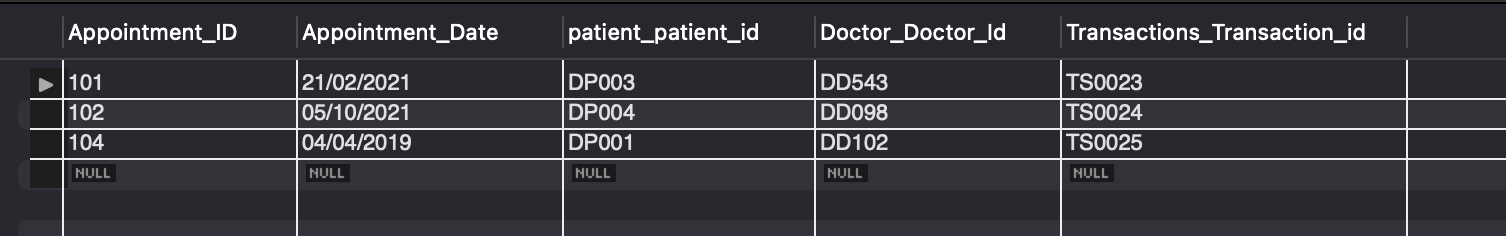
insert into Treatement values('TT1003','teeth setting');

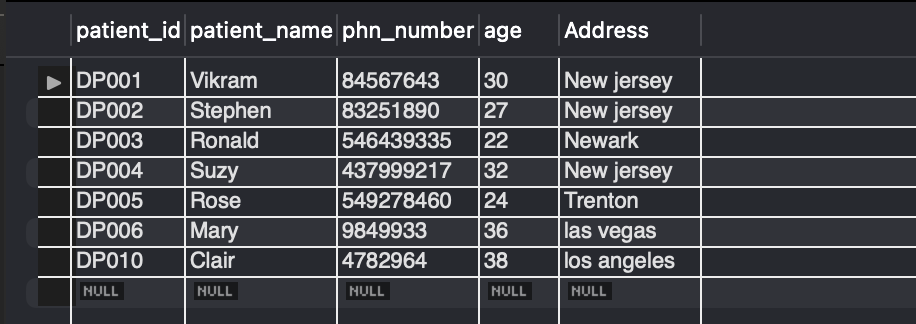
insert into Treatement values('TT1004','Cavity fillings');

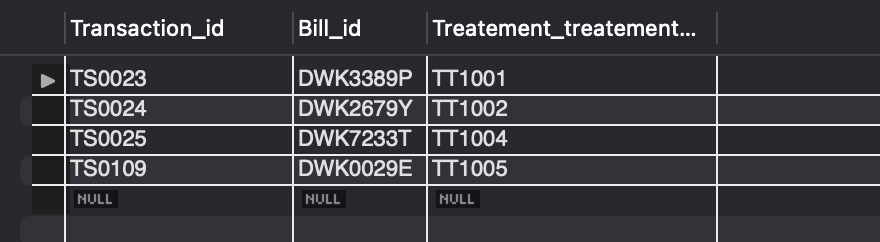
insert into Treatement values('TT1005','Gum Disease Treatment');

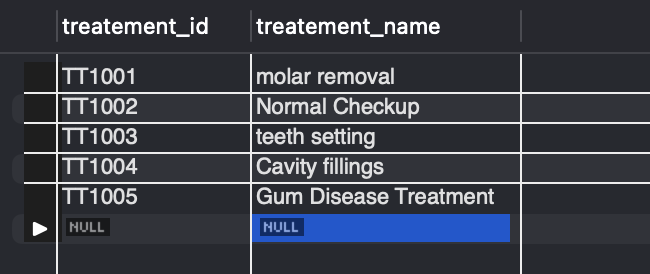












# Data Manipulation Language Scripts

*Description: Write the SQL commands for twelve queries. Two queries should be insert statements, two should update statements, one should be a delete statement, one should be a simple select statement that selects a subset of the rows and columns from one table, two should be a select statements that select data from a joining of two tables, two should use summary functions to generate statistics about the data, one should be a multi-table query, and one should be another query of your choice. Show the queries and screenshots of the results in your Word document, and save your queries in a commented sql script to GitHub.*

*Rubric: Your work will be graded as follows:*

* *1 point each for the two insert statements*
* *1 point each for the two update statements*
* *1 point for the delete statement*
* *1 point for the simple select statement*
* *2 points each for the 2 join statements*
* *2 points each for the two that use summary statements*
* *2 points for the multi-table query*
* *2 points for the query of your choice.*
* *12 points for showing the query and a screenshot of the corresponding result set back-to-back for each of these queries in your Word document.*

*Total points possible: 30*

ENTER DML WORK HERE

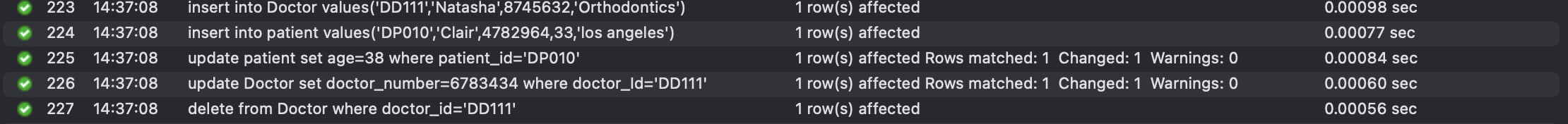
insert into Doctor values('DD111','Natasha',8745632,'Orthodontics');

insert into patient values('DP010','Clair',4782964,33,'los angeles');

update patient set age=38 where patient\_id='DP010';

update Doctor set doctor\_number=6783434 where doctor\_Id='DD111';

delete from Doctor where doctor\_id='DD111';



select \* from Treatment;

Table

Description automatically generated

select a.patient\_patient\_id, p.patient\_name

from patient p

inner join Appointment a on a.patient\_patient\_id = p.patient\_id;

Table

Description automatically generated

select a.Doctor\_Doctor\_Id, d.Doctor\_name

from Doctor d

inner join Appointment a on a.Doctor\_Doctor\_Id= d.Doctor\_Id;

Table

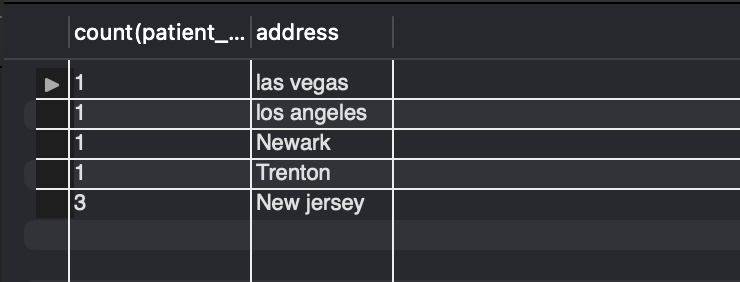
Description automatically generated

Select count(patient\_id),address

from patient

group by address

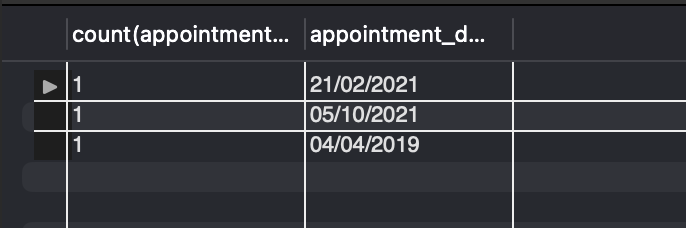
order by count(patient\_id);



select count(appointment\_id), appointment\_date

from Appointment

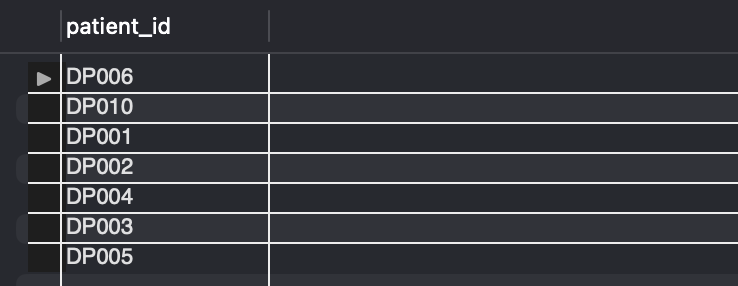
group by appointment\_date;



select patient\_id from patient

union

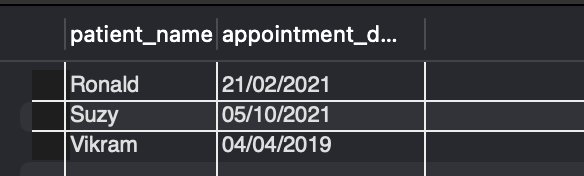
select patient\_patient\_id from Appointment;



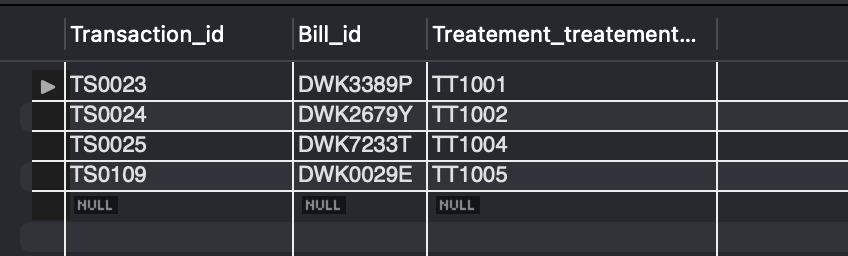
select p.patient\_name, a.appointment\_date

from patient p, Appointment a

where p.patient\_id=a.patient\_patient\_id;



select \* from Transactions;



# Indexes

*Description: Improve the performance of your design by adding indexes to various tables. Show the SQL needed to add the indexes. Explain why you chose the ones you added. Explain how you would demonstrate the impact the indexes had on the performance of various queries.*

*Rubric: Your work will be graded as follows:*

* *6 points for clearly defining at least three indexes and explaining why you chose them.*
* *3 points for showing the sql needed to generate the indexes*
* *3 points for explaining how you would demonstrate the performance improvement afforded by the indexes.*

*Total points possible: 12*

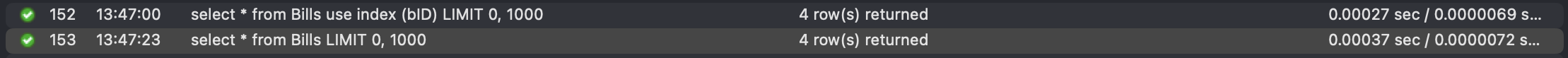
ENTER YOUR INDEX WORK HERE

create index billID on Bills(bill\_id);

create index TranID on Transactions(transaction\_id);

create index PatientADD on Patient(Address);

The reason that I’ve chosen these indexes is because many times the bills and transactions are checked to validate and other reasons and also to access the patients who paid those bills. Moreover, to check how many patients are from a particular area I’ve created the patientADD indexes.



As you can see, the execution time using index is less than without index and with more time on our hand we can focus on the things rather than wait till the records are fetched.

# Views

*Description: Add two views to your database to provide easy access to combinations of data from multiple tables.*

*Rubric: Your work will be graded as follows:*

* *2 points for including the SQL for generating the two views in your Word document*
* *2 points for including screenshots for the data contained in each view in your Word document*
* *2 points for explaining why each view is a valuable addition to your database*
* *2 points for explaining who might benefit most from having access to each view.*

*Total points possible: 8*

ENTER YOUR WORK WITH VIEWS HERE

create view [Patient\_records] as  
select p.patient\_id, p.patient\_name, d.Doctor\_name, b.Bill\_amount  
from patient p, Doctor d, Transactions t, Bills b  
where p.patient\_id=t.patient\_patient\_id and d.Doctor\_Id=t.Doctor\_Doctor\_Id and b.Bill\_id=t.Bill\_id;

create view [Patient\_treatments] as  
select p.patient\_name, tt.treatement\_name  
from patient p, Transactions ts, Treatement tt  
where p.patient\_id=ts.patient\_patient\_id and ts.Treatement\_treatement\_id=tt.treatement\_id;

Graphical user interface

Description automatically generated

The reason for the 1st view is that people tend to view their expenses a lot. Hence, I’ve created that view.

While the second view helps the administrative department monitor the different patients undergoing the treatment.

# Triggers

*Description: Add a trigger to a table so that data will be updated when a certain event occurs*

*Rubric: Your work will be graded as follows:*

* *2 points for including the SQL for the trigger in your Word document*
* *2 points for clearly explaining the purpose of the trigger*
* *2 points for a screenshot and explanation that shows the trigger in action.*

*Total points possible: 6*

ENTER YOUR WORK WITH TRIGGERS HERE

# Transactions

*Description: Demonstrate that you know how to define and use a transaction. Why are transactions important for ensuring ACID behavior?*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly explaining the importance of transactions to ensuring ACID behavior*
* *3 points for including a screenshot and accompanying explanation of a MySQL transaction.*

*Total points possible: 6*

ENTER YOUR WORK WITH TRANSACTIONS HERE

ACID (Atomicity, Consistency, Isolation, and Durability) is a collection of database transaction attributes designed to ensure transaction validity even in the face of errors, power outages, and other problems. A transaction is a set of database activities that meet the ACID criteria and can thus be viewed as a single logical operation on the data.

Atomicity : In every condition, including power outages, malfunctions, and crashes, all statements in a transaction must complete successfully or utterly fail.

Consistency : After any transaction, the database must maintain consistency.

Isolation : Isolation ensures that concurrent transaction execution leaves the database in the same state as if the transactions had been completed sequentially.

Durability : Durability ensures that once a transaction is committed, it will remain committed even if the system fails.

Text

Description automatically generated

In this transaction, patient information and doctor information are being inserted into their tables.

# Database Security

*Description: Identify the different kinds of users who will use your database. Write GRANT statements to define the privileges for these different kinds of users.*

*Rubric: Your work will be graded as follows:*

* *6 points for clearly identifying and describing the various kinds of users who will use the databases and identifying and justifying what privileges each should have.*
* *4 points for writing GRANT statements that assign privileges to these different kinds of users.*
* *4 points for demonstrating with screenshots that your GRANT statements do distinguish among different kinds of users in regard to what they can do with the database.*

*Total points possible: 14*

ENTER YOUR WORK WITH DATABASE SECURITY HERE

Different users are :

Receptionist – the front office worker who manages patients details and makes appointments for them.

manager – the head officer who oversees the clinic records.

Owner of the clinic – the founder of the clinic/hospital

GRANT INSERT, SELECT, UPDATE ON dental\_mngmnt\_sys TO ‘rep\_front’@’localhost’;

GRANT ALL PRIVILEGES ON dental\_mngmnt\_sys TO ‘manager’@’localhost’

GRANT ALL PRIVILEGES ON \*.\* TO ‘owner’@’localhost’

Text

Description automatically generated

# Locking and Concurrent Access

*Description: Explain the purpose of locking tables and show how to do that to prevent inconsistencies that may arise in your data when concurrent transactions take place.*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly explaining an example that shows why you should lock tables to prevent inconsistencies.*
* *3 points for providing a screenshot and accompanying explanation of locking tables.*

*Total points possible: 6*

ENTER YOUR WORK WITH LOCKING AND CONCURRENT ACCESS HERE

Lets say two different people came to dental check-up and took appointment simultaneously. So during the record insertion the appointment records will have the same appointment id causing inconsistencies in the table. Therefore it is better to lock tables during any operation being done on it.

Graphical user interface, text

Description automatically generated

Graphical user interface, application, website

Description automatically generated

# Backing Up Your Database

*Description: How you will back up your database. What commands will you issue? How frequently will the commands run? How can they be automated? Where will the backups be stored?*

*Rubric: Your work will be graded as follows:*

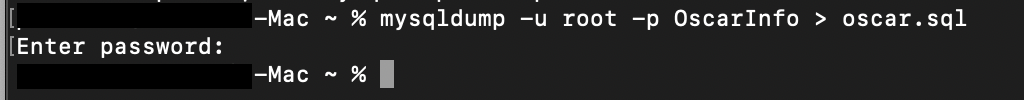
* *12 points for clearly explaining and justifying your database backup strategy, including the frequency with which you will back up the database, how you will automate backups, where you will store them, and how you will secure them. You will earn three points for addressing each factor (frequency, location, automation, and security)*
* *3 points for providing a screenshot of the command you would issue to back up the database and for including a portion of the resulting file.*

*Total points possible: 15*

ENTER YOUR WORK ON DATABASE BACKUPS HERE

The backup frequency that I would choose is once every three days since I would like to go through the records every three days and resolve any issues if arised and the location I choose to store it is on cloud since if the computer crashes I can still download the backup database.

I would automate the database back up by creating a .bat script file and scheduling it using the windows scheduler and I will secure them using the encryption algorithms.



Graphical user interface, text, application

Description automatically generated

# Python Programming

*Description: Write a Python program that generates a report that contains a subset of the data from your database. Include the code for your Python program in your Word document, and also post the program to your GitHub repository.*

*Rubric: Your work will be graded as follows:*

* *12 points for writing a Python script (and including its code in the Word doc) that will pull data from a database and store it to a text file and present it to the screen. Your code must have comments in it that explain how it works. You will be awarded 3 points for successfully connecting to the database, 3 points for successfully querying it, and 4 points for presenting the data to the screen and to a file. Internal comments count for 2 points.*
* *2 points for posting the code to GitHub*
* *4 points for showing a screenshot of your running the script and showing the results it produces on the screen.*

*Total points possible: 18*

ENTER YOUR PYTHON DATABASE PROGRAMMING WORK HERE

import mysql.connector

from mysql.connector import errorcode

from mysql.connector.errors import DataError, Error

table = ['Appointment', 'Bills', 'Doctor', 'patient', 'Transactions', 'Treatment']

user = 'root'

pswrd = 'mysql@123'

host = 'localhost'

database = 'dental\_mngmnt\_sys'

# Trying to connect to database

try:

con = mysql.connector.connect(

*user*=user,

*password*=pswrd,

*host*=host,

*database*=database

)

print("Database connection successfull")

# opening the file

file= open("Tables.txt","w")

print("File open successfull")

cursor = con.cursor()

# fetching the records

for ta in table:

cursor.execute("select \* from "+ta)

print(ta+" Information")

file.write(ta+" Information\n")

for rec in cursor.fetchall():

msge=""

for r in rec:

msge+=str(r)+"\t"

msge+="\n"

# writing the records to the txt file

file.write(msge)

print(msge)

file.write("\n")

print("\n")

except Error as err:

print("error connecting database or opening file")

print(err)

exit()

finally:

# finally closing the connection

con.close()

file.close()

Text

Description automatically generated

Text

Description automatically generated

# PHP Programming or Java Program

*Description: Either*

1. *Build an HTML form that enables the user to specify criteria to search by. Use PHP to show the results of the query on a resulting web page. Make sure you include protections against an SQL injection attack. Include your HTML and PHP code in your Word document, and also post the files to your GitHub repository.*

*Rubric: Your work will be graded as follows:*

* + *4 points for writing an HTML form the user will use to enter search criteria*
  + *8 points for a PHP script that uses the search criteria and returns results*
  + *4 points for an HTML page that shows the results*
  + *4 points for explaining what SQL injection might be run on your website and explaining how you prevented it.*
  + *4 points for providing screen shots of your PHP website in action.*
  + *2 points for posting your code to GitHub*

*or*

1. *Build a Java program that manages the content of your database. It should be a menu-driven program that gives the user the option to insert a new record into one of your tables, update a record from one of your tables, delete a record from one of your tables, or select records from one of your tables according to multiple search criteria the user enters.*

*Rubric: Your work will be graded as follows:*

* + *4 points for repeatedly showing the menu of options until the user decides to quit*
  + *4 points for implementing the insert operation*
  + *4 points for implementing the update operation*
  + *4 points for implementing the delete operation*
  + *8 points for implementing the select operation with multiple criteria*
  + *2 points for posting your code to GitHub*

*Total points possible: 26*

ENTER YOUR PHP or JAVA DATABASE APP PROGRAMMING WORK HERE

import java.sql.\*;

import java.util.\*;

public class JavaConn{

public static *void* main(String *args*[])throws SQLException, ClassNotFoundException {

Scanner sc = new Scanner(System.in);

Connection con = null;

try{

Class.forName("com.mysql.cj.jdbc.Driver");

con=DriverManager.getConnection(

"jdbc:mysql://localhost:3306/dental\_mngmnt\_sys?useSSL=false","root","root@123");

Statement stmt=con.createStatement();

while(con!=null) {

System.out.println();

System.out.println("1.Insert operation");

System.out.println("2.Update operation");

System.out.println("3.Delete operation");

System.out.println("4.Select operation");

System.out.println("5.Exit");

System.out.println("Your choice?");

*int* ch= sc.nextInt();

sc.nextLine();

switch(ch) {

case 1 :{

System.out.println("Enter the table name you want to insert into");

String table = sc.nextLine();

if(table.equalsIgnoreCase("Appointment")){

System.out.println("Enter appointment id, date, patient id, doctor id, transaction id");

String rec[]=sc.nextLine().split(" ");

String record = "insert into Appointment values('"+rec[0]+"','"+rec[1]+"','"+rec[2]+"','"+rec[3]+"','"+rec[4]+"')";

*int* x = stmt.executeUpdate(record);

System.out.println(x>0?"insert successfull!":"insert unsuccessfull!");

}

else if(table.equalsIgnoreCase("Bills")){

System.out.println("Enter bill id, amount, transaction id");

String rec[]=sc.nextLine().split(" ");

String record = "insert into Bills values('"+rec[0]+"','"+rec[1]+"','"+rec[2]+"')";

*int* x = stmt.executeUpdate(record);

System.out.println(x>0?"insert successfull!":"insert unsuccessfull!");

}

else if(table.equalsIgnoreCase("Doctor")){

System.out.println("Enter doctor id, name, number, specialization");

String rec[]=sc.nextLine().split(" ");

String record = "insert into Doctor values('"+rec[0]+"','"+rec[1]+"','"+rec[2]+"','"+rec[2]+"')";

*int* x = stmt.executeUpdate(record);

System.out.println(x>0?"insert successfull!":"insert unsuccessfull!");

}

else if(table.equalsIgnoreCase("Patient")){

System.out.println("Enter patient id, name, phone number, age, address");

String rec[]=sc.nextLine().split(" ");

String record = "insert into Patient values('"+rec[0]+"','"+rec[1]+"','"+rec[2]+"','"+rec[3]+"','"+rec[4]+"')";

*int* x = stmt.executeUpdate(record);

System.out.println(x>0?"insert successfull!":"insert unsuccessfull!");

}

else if(table.equalsIgnoreCase("Transaction")){

System.out.println("Enter Transaction id, bill id, treatment id");

String rec[]=sc.nextLine().split(" ");

String record = "insert into Transaction values('"+rec[0]+"','"+rec[1]+"','"+rec[2]+"')";

*int* x = stmt.executeUpdate(record);

System.out.println(x>0?"insert successfull!":"insert unsuccessfull!");

}

else if(table.equalsIgnoreCase("Treatment")){

System.out.println("Enter Treatment id, treatment name");

String rec[]=sc.nextLine().split(" ");

String record = "insert into Treatment values('"+rec[0]+"','"+rec[1]+"')";

*int* x = stmt.executeUpdate(record);

System.out.println(x>0?"insert successfull!":"insert unsuccessfull!");

}

break;

}

case 2 :{

System.out.println("Please enter the treatment id and the name that you want to change");

String id[]=sc.nextLine().split(" ");

//String name = sc.nextLine();

String record="UPDATE Treatment set treatement\_name='"+id[1]+"' where treatement\_id = '"+id[0]+"'";

*int* x = stmt.executeUpdate(record);

System.out.println(x>0?"Update successfull!":"Update unsuccessfull!");

break;

}

case 3 :{

System.out.println("Please enter the treatment id that you want to delete");

String id=sc.nextLine();

String record="delete from Treatment where treatement\_id = '"+id+"'";

*int* x = stmt.executeUpdate(record);

System.out.println(x>0?"delete successfull!":"delete unsuccessfull!");

break;

}

case 4 :{

String query = "select a.Appointment\_ID, a.Appointment\_Date, p.patient\_name, p.phn\_number, p.Address from Appointment a, Patient p where p.patient\_id=a.patient\_patient\_id";

ResultSet rs=stmt.executeQuery(query);

System.out.println("Appointment\_ID Appointment\_Date Name phn\_number Address");

while(rs.next())

System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3)+" "+rs.getInt(4)+" "+rs.getString(5));

break;

}

case 5: System.exit(0);

}

}

}

catch(SQLException e){

System.out.println(e);

}

catch(ClassNotFoundException e) {

System.out.println(e);

}

finally{

con.close();

sc.close();

}

}

}

Text

Description automatically generated

# Suggested Future Work

*Description: Describe the limitations of your current database and explain how you or someone else could improve the design to address these shortcomings. Also describe how you might take advantage of leverage cloud services to increase the performance and availability of your database. Finally, explain the advantages and disadvantages of storing your data in a NoSQL format instead.*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly describing the limitations of your databases*
* *3 points for explaining how you would address these shortcomings*
* *3 points for explaining how you might migrate the database to the cloud and describing what advantages you might gain from doing that.*
* *3 points for explaining the advantages and disadvantages of storing your data in a document-based NoSQL format instead.*

*Total points possible: 12*

ENTER YOUR SUGGESTED FUTURE WORK IDEAS HERE

1. The main limitations that this database has is that it doesn’t include the remaining working staff and the actions they perform and also the data integrity is slightly compromised.
2. To resolve these issues, I would introduce an additional table called staff which holds only the staff’s data and also create some views which helps in better performance.
3. Migrating this database to the cloud has many advantages such as the low operational costs and proper end to end encryption.
4. The advantages of having this database as a document-based include faster performance and better scalability while the disadvantages include lack of standardization and compatibility issues with the sql instructions.

# Activity Log

*Description: As an appendix, the team will keep a daily diary or log of their activity. What did you or your team study in this class each day? What did you learn? What did you accomplish or build or design? You don't have to enter something every day, but there should be at least three entries each week. Since we have eight weeks, that means you should make 3 posts to the Activity Log each week, for a total of at least 24 posts. Each post will be worth 1 point.*

*If you are working as part of a team, make sure you clearly identify which team member worked on which tasks. The Activity Log should help me figure out how each team member contributed to the project. If I cannot discern who worked on what aspects of the project from the activity log, no points will be awarded for it.*

*Total points possible: 24*

MAKE AT LEAST THREE ENTRIES PER WEEK. CLEARLY IDENTIFY WHAT EACH PERSON ON YOUR TEAM ACCOMPLISHED. YOU MUST SHARE THE RESPONSIBILITY OF COMPLETING THE PROJECT.