**Lewis University  
 CPSC 50900: Database Systems   
 Spring 2023 Term Project**

**STUDENT RECORD KEEPING SYSTEM DATABASE PROJECT**

**Mallikarjun Ghanapuram , Mallikarjunghanapu@lewisu.edu**

Work products stored in the Github repository [mallikarjunghanapuram/database-assignment (github.com)](https://github.com/mallikarjunghanapuram/database-assignment)

Table of Contents

[Initial Proposal 2](#_Toc133354985)

[Data Sources 2](#_Toc133354987)

[Alternative Ways to Store the Data 3](#_Toc133354989)

[Relational Database Design Process 4](#_Toc133354991)

[Relational Database Design 6](#_Toc133354993)

[Data Definition Language (DDL) Scripts 9](#_Toc133354994)

[Data Manipulation Language Scripts 12](#_Toc133354995)

[Indexes 17](#_Toc133354996)

[Views 20](#_Toc133354997)

[Triggers 22](#_Toc133354998)

[Transactions 24](#_Toc133354999)

[Database Security 25](#_Toc133355000)

[Locking and Concurrent Access 27](#_Toc133355001)

[Backing Up Your Database 27](#_Toc133355002)

[Programming 29](#_Toc133355003)

[Suggested Future Work 30](#_Toc133355004)

[Activity Log 32](#_Toc133355005)

# 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*

# **Initial Proposal**

A student record or data set includes details specifically pertaining to a particular student, such as the student's name, student ID, address, guardian information, medical information, room number, name, and associated passport photo, as well as a list of personal attributes.

To manage the relationship between the institution and the student, control the student's academic development, and evaluate their performance, an institution must maintain records pertaining to its students.

Student records also contain information that the institution can compile and evaluate in order to inform future strategy, planning, and service delivery.

The application is distributed because administrators and academic staff from all throughout the institution can access the data, which is kept centrally by the university. Students can also view their results by logging into the system. As the real system would be service-oriented, part of its functionality would be made available as a collection of services that other applications may use.

# 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*

# **Data Sources**

The sources of a student record management system include:

1. Student record management
2. Admission management
3. Attendance management
4. Fee management

You can manage student data to keep up with all school-related information using a student record system's unified database. This makes it easier for you to read important information on admission, courses, and syllabuses. In order to examine learning outcomes, grades, performance, attendance, and other factors, you can also create custom reports using the student records database. You may therefore make better educated decisions because you have a complete picture of your kids.

# Alternative Ways to Store the Data

*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*

# **Data Storage Alternatives**

Users now have the option to exchange and access files remotely without having to use their local storage systems thanks to cloud storage.

Advantages of cloud storage are:

Cloud storage is significantly less expensive per GB than using external drives without the requirement for hardware.

Accessibility: When you save your files in the cloud, you may access them from any location with an internet connection.

Recovery: You can access your files in the cloud in case of a hard drive crash or other technical issue. It serves as a backup option for the physical drives that make up your local storage.

Updating and Syncing: Every time you make changes to a file while using cloud storage, those changes are synced and updated across all of the devices you use to access the cloud.

Security: Cloud storage companies increase the security of their services.

Disadvantages of cloud storage are:

Internet accessibility: The use of cloud storage requires an internet connection. You can experience difficulties gaining access to your storage if your network is slow. You won't be able to view your files if you are somewhere without internet connectivity.

Costs: The charges associated with downloading and uploading data to the cloud are extra. If you frequently attempt to access several files, these costs might pile up very rapidly.

Disk drives: Our reliance on hard drives should be eliminated thanks to cloud storage, right? Well, some commercial cloud storage companies also demand physical hard drives.

The greatest cloud storage support isn't available, especially if you're using a free cloud provider. Several providers suggest visiting a knowledge base.

# 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*

# **Relational Database Design Process**

We can identify the following functional dependencies:

Admission table:

Determinant: Admission ID

Dependent Fields: Admission Date, Admission Status, Phone

Attendance table:

Determinant: Attendance Date, Student ID

Dependent Fields: Student Present, Student Absent

Payment table:

Determinant: Receipt Number

Dependent Fields: Student Name, Amount Paid, Student Phone

Student table:

Determinant: Student ID

Dependent Fields: Student Name, Student Class, Student Phone

Based on the functional dependencies, following entity sets or tables can be identified:

Admission (with Admission ID as the primary key)

Attendance (with a composite primary key of Attendance Date and Student ID)

Payment (with Receipt Number as the primary key)

Student (with Student ID as the primary key)

The relationships among the entity sets can be identified as follows:

Admission and Student have a one-to-one relationship (one admissions can belong to one student)

Attendance and Student have a many-to-one relationship (many attendance records can belong to one student)

Payment and Student have a many-to-one relationship (many payments can be made by one student)

For each relationship, we can identify the connectivity and participation as follows:

Admission to Student: one-to-one, mandatory (one admission must belong to a student)

Attendance to Student: many-to-one, mandatory (every attendance record must belong to a student)

Payment to Student: many-to-one, mandatory (every payment must be made by a student)

# 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 Vertabelo. 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 Vertabelo 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*

**Relational Database Design:**

As following entities are be identified:

Admission

Admission\_ID (Primary Key, INT)

Admission\_Date (DATE)

Admission\_Status (VARCHAR)

Phone (VARCHAR)

Student\_ID (INT)

Attendance

Attendance\_ID (Primary Key, INT)

Attendance\_Date (DATE)

Student\_ID (Foreign Key, INT)

Student\_Present (VARCHAR)

Student\_Absent (VARCHAR)

Payment

Receipt\_Number (Primary Key, INT)

Student\_ID (Foreign Key, INT)

Amount\_Paid (DECIMAL)

Payment\_Date (DATE)

Student

Student\_ID (Primary Key, INT)

Student\_Name (VARCHAR)

Student\_Class (VARCHAR)

Student\_Phone (VARCHAR)

To ensure that the tables are in 2nd Normal Form, we have ensured that each table has a single primary key attribute. All other attributes are dependent on this primary key attribute. Therefore, each table is in 2nd Normal Form.

To ensure that the tables are in 3rd Normal Form, we need to check that there are no transitive dependencies. We can see that the Student table has a transitive dependency between Student\_ID, Student\_Name, Student\_Class, and Student\_Phone. To remove this transitive dependency, we can create a new table called Student\_Details as follows:

Student\_Details

Student\_ID (Primary Key, INT)

Student\_Name (VARCHAR)

Student\_Phone (VARCHAR)

Student

Student\_ID (Primary Key, INT)

Student\_Class (VARCHAR)

Student\_Details\_ID (Foreign Key, INT)

Now, Student\_Details has Student\_ID as its primary key and Student\_Name and Student\_Phone as dependent attributes. Student table has Student\_ID as its primary key, Student\_Class as a dependent attribute, and a new foreign key Student\_Details\_ID. This ensures that each table is in 3rd Normal Form.

To ensure that the tables are in Boyce-Codd Normal Form (BCNF), we need to ensure that every determinant is a candidate key. We can see that all tables already have a single primary key attribute, which is a candidate key. Therefore, all tables are in BCNF.

We also need to introduce foreign keys to connect related tables. We have already added foreign keys to Attendance, Payment, and Student tables.

Admission\_ID (Foreign Key, INT)

Attendance\_ID (Foreign Key, INT)

To make 8 entities we can add course, enrolment and professor table.

Course

Course\_ID (Primary Key, INT)

Course\_Name (VARCHAR)

Enrolment

Course\_ID(Foreign Key, INT)

Student\_ID(Foreign Key, INT)

Professor

Prof\_ID (Primary Key, INT)

Prof\_Name (VARCHAR)

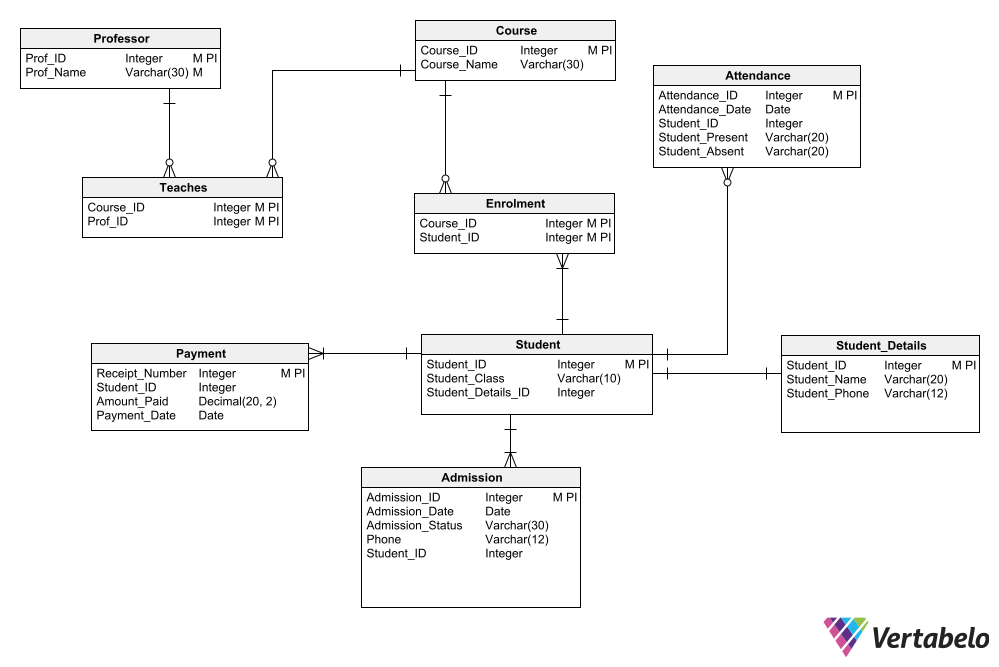
One bridge entity can be added as:

Teaches

Course\_ID(Foreign Key, INT)

Prof\_ID(Foreign Key, INT)

Here is the ER diagram:



# 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 Vertabelo 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*

**Data Definition Language (DDL) Scripts:**

Sql commands to populate the tables:

-- Insert three records into the Student table

INSERT INTO Student (Student\_ID, Student\_Class, Student\_Details\_ID)

VALUES (101, '10A', 1),

(102, '10B', 2),

(103, '10C', 3);

-- Insert three records into the Student\_Details table

INSERT INTO Student\_Details (Student\_ID, Student\_Name, Student\_Phone)

VALUES (1, 'Alice Brown', '555-1234'),

(3, 'Bob Green', '555-5678'),

(2, 'Charlie Lee', '555-9012');

-- Insert three records into the Course table

INSERT INTO Course (Course\_ID, Course\_Name)

VALUES (1, 'Mathematics'),

(2, 'English'),

(3, 'Science');

-- Insert three records into the Admission table

INSERT INTO Admission (Admission\_ID, Admission\_Date, Admission\_Status, Phone, Student\_ID)

VALUES (1, '2022-09-01', 'Accepted', '555-1234', 101),

(2, '2022-09-01', 'Accepted', '555-5678', 102),

(3, '2022-09-02', 'Pending', '555-9012', 103);

-- Insert three records into the Attendance table

INSERT INTO Attendance (Attendance\_ID, Attendance\_Date, Student\_ID, Student\_Present, Student\_Absent)

VALUES (1, '2022-09-01', 101, 'Yes', NULL),

(2, '2022-09-01', 102, NULL, 'Yes'),

(3, '2022-09-02', 103, 'Yes', NULL);

-- Insert three records into the Enrolment table

INSERT INTO Enrolment (Course\_ID, Student\_ID)

VALUES (1, 101),

(2, 102),

(3, 103);

-- Insert three records into the Payment table

INSERT INTO Payment (Receipt\_Number, Student\_ID, Amount\_Paid, Payment\_Date)

VALUES (1, 101, 1000.00, '2022-09-05'),

(2, 102, 500.00, '2022-09-06'),

(3, 103, 800.00, '2022-09-07');

-- Insert three records into the Professor table

INSERT INTO Professor (Prof\_ID, Prof\_Name)

VALUES (1, 'John Smith'),

(2, 'Mary Johnson'),

(3, 'David Lee');

-- Insert three records into the Teaches table

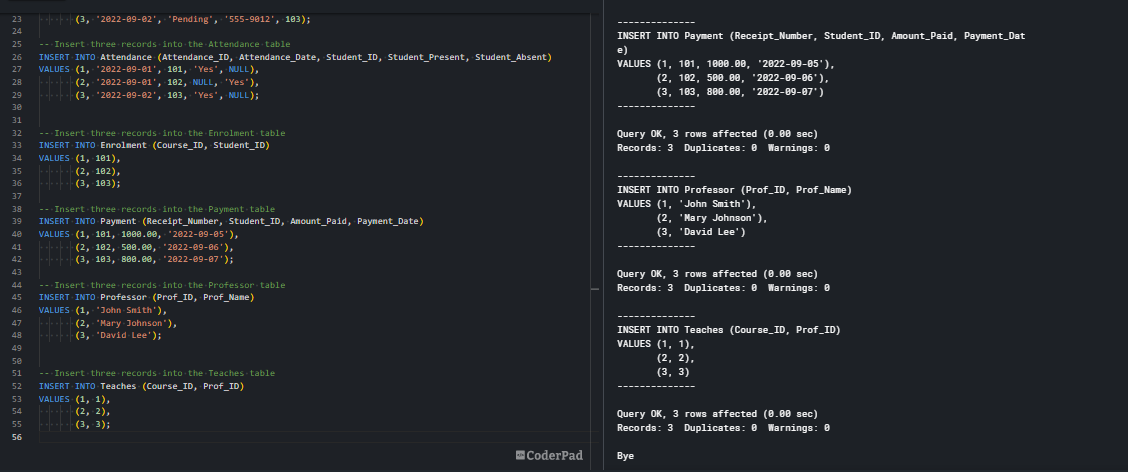
INSERT INTO Teaches (Course\_ID, Prof\_ID)

VALUES (1, 1),

(2, 2),

(3, 3);

Screenshot for execution statements for populating records;



# 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.*
* *6 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: 24*

**Data Manipulation Language Scripts:**

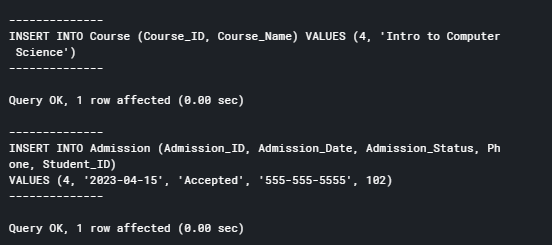
* Two insert statements are:

INSERT INTO Course (Course\_ID, Course\_Name) VALUES (4, 'Intro to Computer Science');

INSERT INTO Admission (Admission\_ID, Admission\_Date, Admission\_Status, Phone, Student\_ID)

VALUES (4, '2023-04-15', 'Accepted', '555-555-5555', 102);

Screenshot of the output:

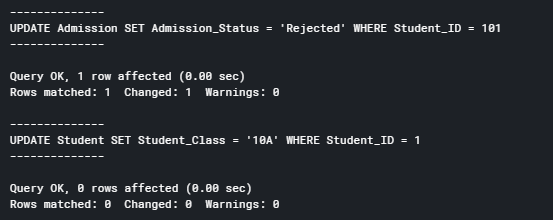


* Two update statements are:

UPDATE Admission SET Admission\_Status = 'Rejected' WHERE Student\_ID = 101;

UPDATE Student SET Student\_Class = '10A' WHERE Student\_ID = 1;

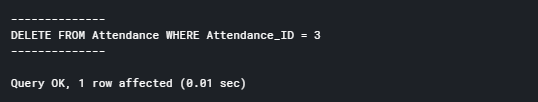
Output:



* One delete statement:

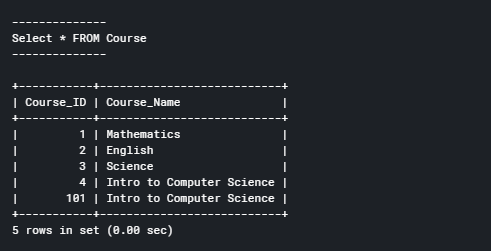
DELETE FROM Attendance WHERE Attendance\_ID = 3;

Output:



* Select \* FROM Course;

Output:



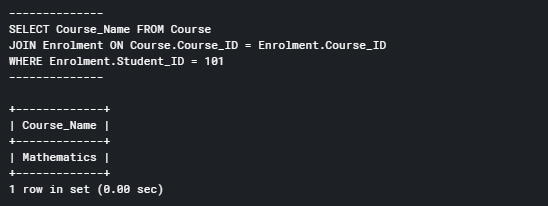
* 2 join statements are:

SELECT Course\_Name FROM Course

JOIN Enrolment ON Course.Course\_ID = Enrolment.Course\_ID

WHERE Enrolment.Student\_ID = 101;

Output:



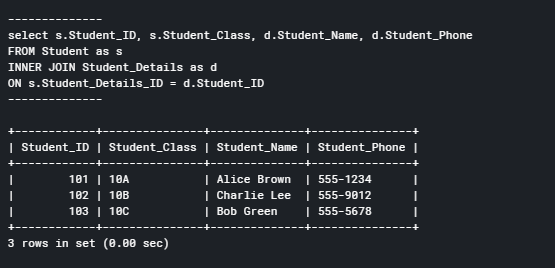
SELECT s.Student\_ID, s.Student\_Class, d.Student\_Name, d.Student\_Phone

FROM Student as s

INNER JOIN Student\_Details as d

ON s.Student\_Details\_ID = d.Student\_ID;

Output:

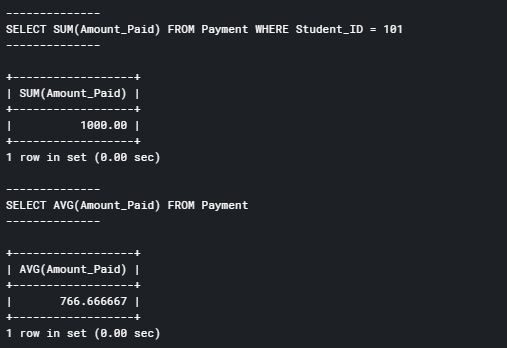


* 2 summary statements are:

SELECT SUM(Amount\_Paid) FROM Payment WHERE Student\_ID = 101;

SELECT AVG(Amount\_Paid) FROM Payment;

Output:



* Multi table query as name of course taught by “John Smith”.

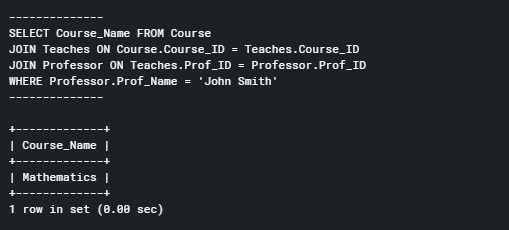
SELECT Course\_Name FROM Course

JOIN Teaches ON Course.Course\_ID = Teaches.Course\_ID

JOIN Professor ON Teaches.Prof\_ID = Professor.Prof\_ID

WHERE Professor.Prof\_Name = 'John Smith';

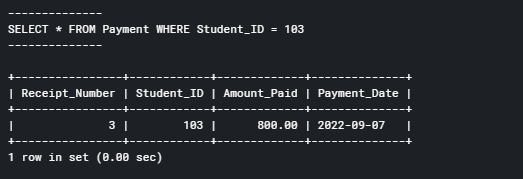
Output:



* A sql query:

SELECT \* FROM Payment WHERE Student\_ID = 103;

Output:



# 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:*

* *3 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*
* *2 points for explaining how you would demonstrate the performance improvement afforded by the indexes.*

*Total points possible: 8*

**Indexes:**

To improve the performance of the database, we can add indexes to the following tables and columns:

Student table:

Index on Student\_Details\_ID column: since this column is used as a foreign key in other tables, an index on it can help improve performance of queries that join these tables.

Enrolment table:

Index on Student\_ID column: since this column is used in several foreign keys and join conditions, an index on it can help speed up these queries.

Teaches table:

Index on Prof\_ID column: since this column is used in several foreign keys and join conditions, an index on it can help speed up these queries.

SQL statements to add these indexes are:

-- Index on Student\_Details\_ID column in Student table

CREATE INDEX Student\_Details\_ID\_idx ON Student (Student\_Details\_ID);

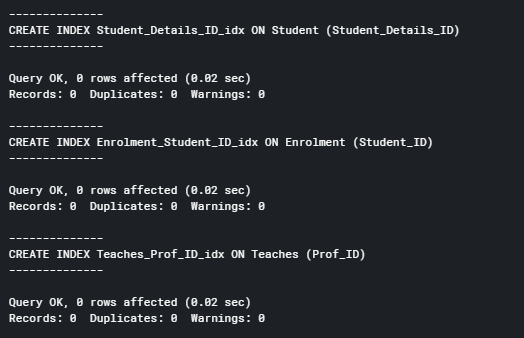
-- Index on Student\_ID column in Enrolment table

CREATE INDEX Enrolment\_Student\_ID\_idx ON Enrolment (Student\_ID);

-- Index on Prof\_ID column in Teaches table

CREATE INDEX Teaches\_Prof\_ID\_idx ON Teaches (Prof\_ID);

Output:



To demonstrate the performance improvement we can following queries before and after the creating indexes:

-- Query 1: Joining Enrolment and Student, Student\_Details tables on Student\_ID column

SELECT e.Course\_ID, sd.Student\_Name

FROM Enrolment e

JOIN Student s ON e.Student\_ID = s.Student\_ID

JOIN Student\_Details sd on s.Student\_Details\_ID = s.Student\_ID

WHERE s.Student\_Class = '10A';

-- Query 2: Joining Teaches, Course, and Professor Tables on Prof\_ID column

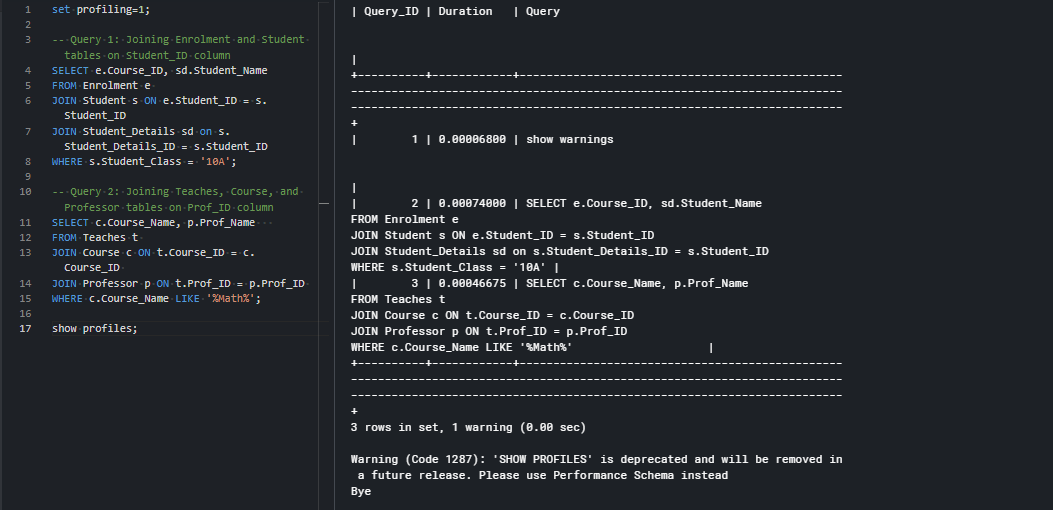
SELECT c.Course\_Name, p.Prof\_Name

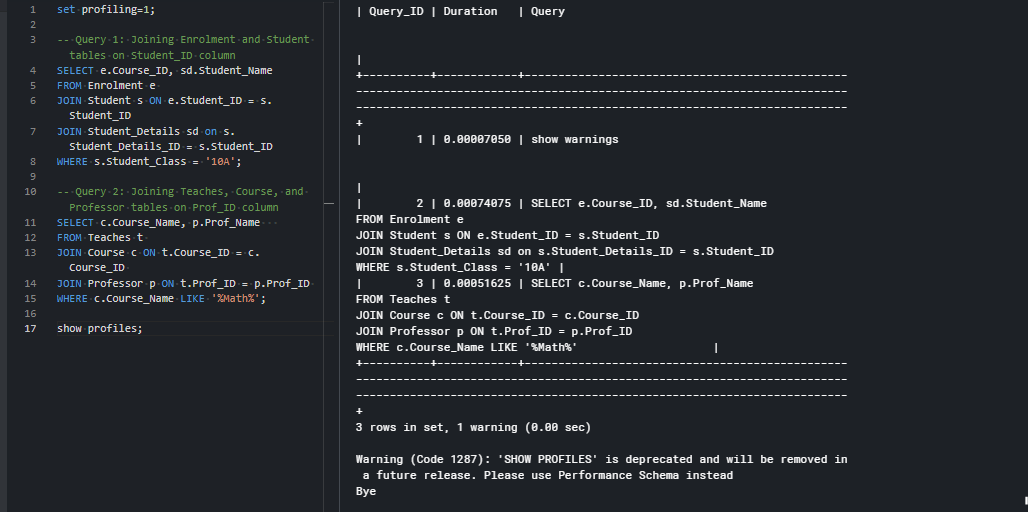
FROM Teaches t

JOIN Course c ON t.Course\_ID = c.Course\_ID

JOIN Professor p ON t.Prof\_ID = p.Prof\_ID

WHERE c.Course\_Name LIKE '%Math%';





# 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*

*Total points possible: 6*

**Views:**

The following two views can be added to the database:

CREATE VIEW Enrolled\_Students AS

SELECT e.Course\_ID, c.Course\_Name, sd.Student\_Name, sd.Student\_Phone, s.Student\_Class

FROM Enrolment e

INNER JOIN Student s ON e.Student\_ID = s.Student\_ID

INNER JOIN Student\_Details sd ON s.Student\_Details\_ID = sd.Student\_ID

INNER JOIN Course c ON e.Course\_ID = c.Course\_ID;

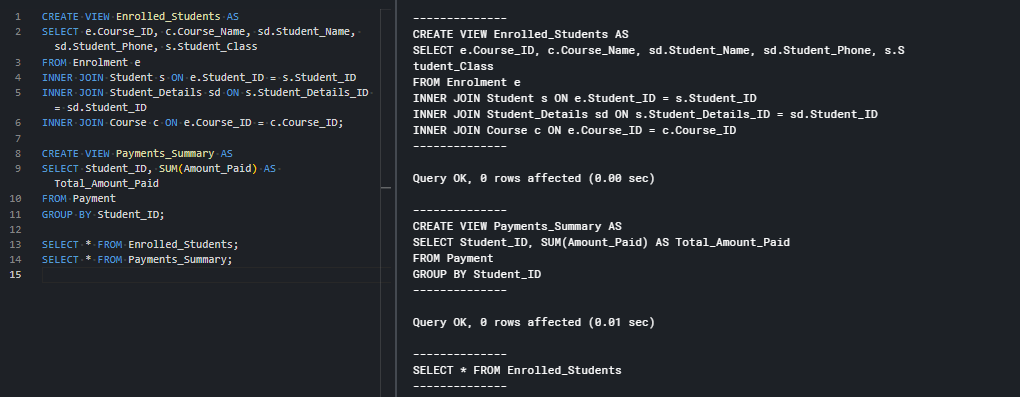
CREATE VIEW Payments\_Summary AS

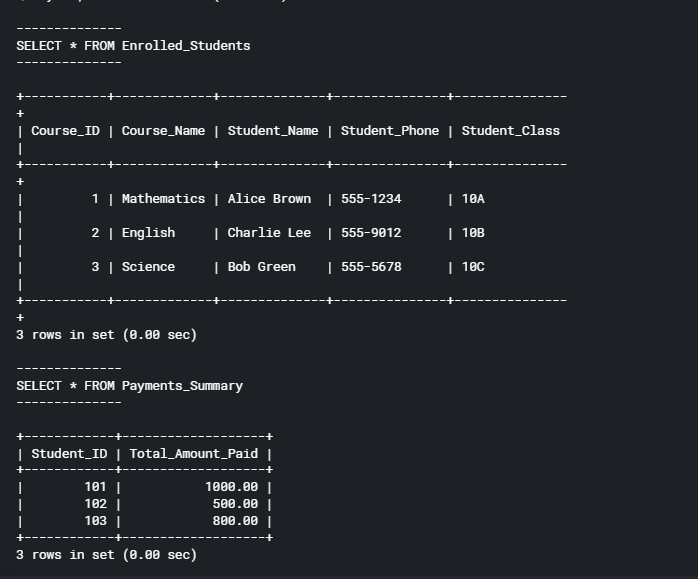
SELECT Student\_ID, SUM(Amount\_Paid) AS Total\_Amount\_Paid

FROM Payment

GROUP BY Student\_ID;

Output:





The Enrolled\_Students view is a valuable addition to the database because it provides a simple way to get information about which students are enrolled in which courses. This can be especially useful for faculty members who need to quickly look up which students are enrolled in their courses.

The Payments\_Summary view is a valuable addition to the database because it provides a summary of the total amount paid by each student. This can be useful for financial aid officers or administrators who need to keep track of payments made by each student. The view allows them to quickly see how much each student has paid without having to run a complex query.

# 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*

**Triggers:**

Here's a trigger that updates the Admission\_Status column in the Admission table when a new record is inserted into the Payment table:

CREATE TRIGGER update\_admission\_status

AFTER INSERT ON Payment

FOR EACH ROW

UPDATE Admission

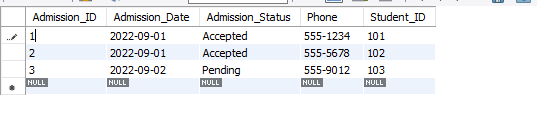
SET Admission\_Status = 'Paid'

WHERE Student\_ID = NEW.Student\_ID;

The purpose of this trigger is to automate the process of updating the admission status of a student when they make a payment. This can save time and reduce errors by eliminating the need for manual updates.

This trigger will fire after a new record is inserted into the Payment table, and for each row affected by the insert, it will update the Admission\_Status column in the Admission table to "Paid" for the corresponding student.

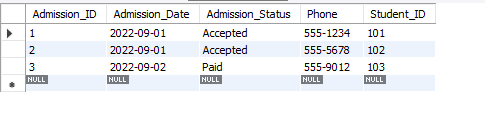
Records of Admission table before payment made by student:



The records of Admission table after inserting Payment records of student by below query:

INSERT INTO Payment (Receipt\_Number, Student\_ID, Amount\_Paid, Payment\_Date)

VALUES (4, 103, 100.00, '2023-04-20');



# 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*

**Transactions:**

Transactions are important for ensuring ACID (Atomicity, Consistency, Isolation, Durability) behavior in database systems. Transactions are a sequence of database operations that are treated as a single unit of work. The ACID properties of transactions ensure that the database is consistent and reliable, even in the face of errors or system failures.

Atomicity ensures that a transaction is treated as a single unit of work, meaning that either all of the changes made by the transaction are committed or none of them are. This prevents partial updates to the database that could leave it in an inconsistent state.

Consistency ensures that a transaction brings the database from one valid state to another. This means that the database will not be left in an invalid state, even in the face of errors or system failures.

Isolation ensures that each transaction is executed independently of other transactions. This means that transactions are executed in a way that does not interfere with each other, which prevents data inconsistencies and other issues.

Durability ensures that once a transaction is committed, the changes made by that transaction will be permanent, even in the face of system failures or other errors.

Below is an example of transaction query in mysql:

START TRANSACTION;

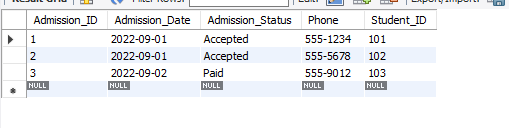
UPDATE Student\_Details SET Student\_Phone = '555-1454' WHERE Student\_ID = 1;

UPDATE Admission SET Phone = '555-1454' WHERE Student\_ID = 1;

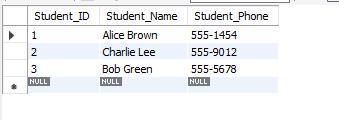
COMMIT;

Output:

Admission table:



Student\_Details table:



# 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: 12*

**Database Security:**

The potential user the database will be :

Administrators: They will have full control over the database, including creating and modifying tables, views, and stored procedures. They will also be able to grant and revoke privileges to other users.

GRANT ALL PRIVILEGES ON . TO 'admin'@'localhost';

Professors: They will be able to view and update the course, enrolment, and teaches tables. They will also be able to view the attendance table for their courses.

GRANT SELECT, UPDATE ON Course TO 'professor'@'localhost';

GRANT SELECT, INSERT, UPDATE, DELETE ON Enrolment TO 'professor'@'localhost';

GRANT SELECT, INSERT, UPDATE, DELETE ON Teaches TO 'professor'@'localhost';

GRANT SELECT ON Attendance TO 'professor'@'localhost';

Students: They will be able to view their admission and payment details, and their enrolment status. They will also be able to view their attendance record.

GRANT SELECT ON Admission TO 'student'@'localhost';

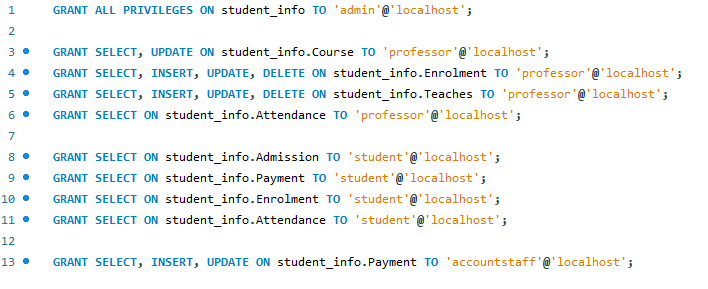
GRANT SELECT ON Payment TO 'student'@'localhost';

GRANT SELECT ON Enrolment TO 'student'@'localhost';

GRANT SELECT ON Attendance TO 'student'@'localhost';

Accounts staff: They will be able to view and update the payment table.

GRANT SELECT, INSERT, UPDATE ON Payment TO 'accountstaff'@'localhost';



# 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: 5*

**Locking and Concurrent Access:**

Locking tables is a technique used to prevent multiple transactions from accessing the same data simultaneously, which can result in inconsistencies in the data. When two transactions try to update the same data at the same time, they may interfere with each other, leading to incorrect or incomplete results. Locking tables allows one transaction to access the data exclusively, preventing other transactions from accessing it until the first transaction completes.

For example, suppose two users, A and B, try to update the same row in the Admission table at the same time. User A begins the transaction and updates the row, but before the transaction is committed, user B begins another transaction and updates the same row. If user B commits the transaction first, user A's changes will be lost, resulting in inconsistent data.

Following queries can be executed to demonstrate locking tables:

LOCK TABLE Admission WRITE;

UPDATE Student SET Student\_Class = '10D' WHERE Student\_ID = 1;

Output:



# 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:*

* *6 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: 8*

**Backing Up Your Database:**

To back up a database, one commonly used method is to use the mysqldump command. This command creates a backup of the database in the form of a SQL file that can be used to restore the database in case of data loss or corruption.

The frequency of the backups will depend on the specific needs of the organization. If the data is constantly changing and needs to be frequently updated, backups can be done on a daily basis. If the data changes less frequently, backups can be done weekly or even monthly. For the student\_info database the backup of the database can be taken weekly.

To automate backups, we can use a cron job in Linux or a scheduled task in Windows to run the backup command at the specified intervals.

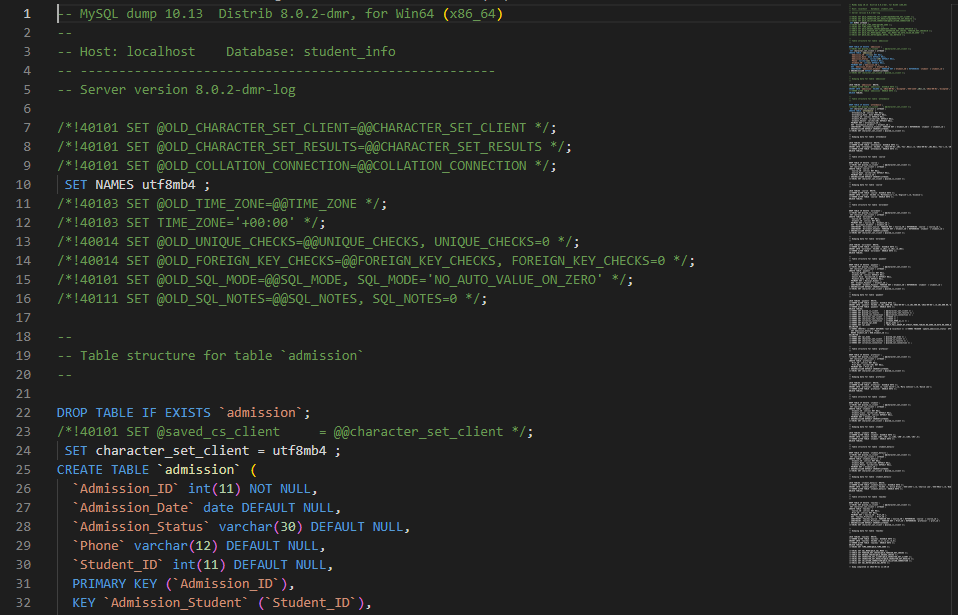
It's important to store backups in a secure location to protect against data loss or theft. Backups can be stored on a separate physical server, in the cloud, or on external hard drives.

To secure the backups, it's important to use encryption and access controls to prevent unauthorized access to the backup files.

Here is the command to create a backup file:

mysqldump -u root -p student\_info > Database\_backup.sql

Output file:



# Programming

*Description: Write a Python, Java, or PHP 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:*

* *10 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*
* *6 points for showing a screenshot of your running the script and showing the results it produces on the screen.*

*Total points possible: 18*

**Programming:**

Here is the script that connect to the database and print name of all students on the screen:

import mysql.connector

# connect to the database

mydb = mysql.connector.connect(

  host="localhost",

  user="root",

  password="toor",

  port = 3306,

  database='student\_info'

)

# create a cursor for acessing data

mycursor = mydb.cursor()

# query all students names in the database

mycursor.execute("SELECT Student\_Name FROM Student\_Details")

# print each student name

for name in mycursor:

  print(name[0])

Output:



# 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*

**Suggested Future Work:**

Limitation of the current schema:

Lack of indexing: Without proper indexing, queries may take longer to execute and database performance could suffer as the size of data grows.

Limited scalability: As the database grows, it may become difficult to manage and scale on a single server.

Limited data security: The schema does not have any provisions for data encryption, which could make the data vulnerable to unauthorized access or modification.

Shortcoming can be addressed as:

indexing: More index can be identified and an added to making data access faster.

Scalability: The schema could be optimized for scalability by using techniques such as sharding, replication, or caching. Alternatively, a NoSQL database could be used instead of a traditional relational database.

Data security: Security features such as encryption, access control, and audit trails could be implemented to protect the data from unauthorized access or modification.

To leverage cloud services, one could consider using a database-as-a-service (DBaaS) platform such as Amazon RDS, Google Cloud SQL, or Microsoft Azure Database. These services provide automatic scalability, high availability, and disaster recovery features, as well as easy integration with other cloud services such as storage, compute, and analytics.

Advantages of using a document-based NoSQL format include:

Flexible data models: NoSQL databases allow for flexible and dynamic data models that can adapt to changing business requirements.

Scalability: NoSQL databases are designed to scale horizontally by adding more nodes to a cluster, which makes them ideal for handling large volumes of data and high traffic loads.

High performance: NoSQL databases can provide faster performance compared to traditional relational databases, especially for read-heavy workloads.

Cost-effective: NoSQL databases can be more cost-effective than relational databases, especially for cloud-based deployments where pricing is based on usage.

Disadvantages of using a document-based NoSQL format include:

Lack of transaction support: NoSQL databases may not support ACID transactions, which could make them unsuitable for certain types of applications.

Limited querying capabilities: NoSQL databases may not provide the same level of query capabilities as relational databases, which could make it harder to retrieve and analyze data.

Data consistency: NoSQL databases may not guarantee strong consistency across all nodes in a cluster, which could lead to data inconsistencies and conflicts.

Learning curve: NoSQL databases require a different mindset and skillset compared to relational databases, which could make it harder to manage and maintain.

# 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*

**Activity log**

13th of march 2023- Reviewed the course project description and selected a project which maintains records of students.

17th of march 2023- Finalized the project title as “**Student record keeping system database project”**

18th of march 2023- Worked on Initial proposal

23nd of march 2023 – Researched and reviewed various options of data maintenance.

25th of march 2023- Researched on various options on data storage.

27thof march 2023- Preparation of report

29thof march 2023 – Worked on Relational Database Design Process

30thof march 2023 – Worked on Relational Database Design

31thof march 2023 – worked on Data Definition Language Scripts

1thof april 2023- Worked on Data Manipulation Language Scripts

2nd of april 2023 - Preparation of report

4thof april 2023 – worked on indexes

7thof april 2023 – worked on views

8thof april 2023 – worked on Triggers

9thof april 2023 – worked on Transactions

13thof april 2023 – worked on Security

15thof april 2023 – worked on Locking

19thof april 2023 – worked on Backup

22thof april 2023 – worked on Python, Java or PHP programming

23thof april 2023 – worked on Suggested Future Work

24thof april 2023 – Preparation of report