



Mini Project

A report submitted to the

Department of Computer Engineering

Faculty of Engineering

University of Ruhuna

Sri Lanka

On 9th of April 2024

In completing an assignment for the module ECE4350 Database Systems

By

Samarasinghe C.Y - EG/2021/4775

Samarasinghe N.A.C.V – EG/2021/4776

Sandaruwan G.G.A - EG/2021/4782

Contents

INTRODUCTION	1
CHAPTER 1 – REQUIREMENT ANALYSIS.....	2
Functional Requirements.....	2
Data Requirements	2
CHAPTER 2 – CONCEPTUAL DESIGN	4
CHAPTER 3 – IMPLEMENTATION	6
Schema Creation.....	6
Table Definitions	6
Insertion	13
Update Operation	19
Deletion	22
CHAPTER 4 – TRANSACTION	23
Simple Queries	23
Complex Queries.....	27
CHAPTER 5 – DATABASE TUNING.....	34
REFERENCES.....	44

Table of Figures

Figure 1 : Main Phases of Databases Designing [1].....	1
Figure 2 : ER Diagram for the Laboratory Management System.....	4
Figure 3 : UML Class Diagram for the Laboratory Management System	5
Figure 4 : Creation of Schema Laboratory Management System.....	6
Figure 5 : Creation of Labs Table	6
Figure 6 : Creation of Lab Maintenance Table	7
Figure 7 : Creation of Lab Equipment Table	7
Figure 8 : Creation of Issues Table	8
Figure 9 : Creation of Lab Status Table	8
Figure 10 : Creation of Location Table.....	9
Figure 11 : Creation of Students Table	9
Figure 12 : Creation of Lab Card Table	10
Figure 13 : Creation of Modules Table.....	10
Figure 14 : Creation of Module Coordinator Table	11
Figure 15 : Creation of Assessments Table.....	11
Figure 16 : Definition of the Derived attribute 'Grade'	12
Figure 17 : Creation of Admins Table	12
Figure 18 : Data Insertion to the Labs Table	13
Figure 19 : Data Insertion to the Lab Maintenance Table	13
Figure 20 : Data Insertion to the Lab Equipment Table	14
Figure 21 : Data Insertion to the Issues Table	14
Figure 22 : Data Insertion to the Lab Status Table	15
Figure 23 : Data Insertion to the Location Table	15
Figure 24 : Data Insertion to the Students Table	16
Figure 25 : Data Insertion to the Lab Card Table	16
Figure 26 : Data Insertion to the Modules Table	17
Figure 27 : Data Insertion to the Module Coordinator Table.....	17
Figure 28 : Data Insertion to the Assessments Table.....	18
Figure 29 : Data Insertion to the Admins Table.....	18
Figure 30 : Data Updates in Labs and Lab Maintenance Tables	19
Figure 31 : Data Updates in Lab Equipment and Issues Tables	19
Figure 32 : Data Updates in Lab Status and Location Tables.....	20
Figure 33 : Data Updates in Students, Lab Card and Modules Tables	20
Figure 34 : Data Updates in Module Coordinator, Assessments and Admins Tables.....	21
Figure 35 : Deleting data from each table	22
Figure 36 : Project Operation	23
Figure 37 : Select Operation.....	23
Figure 38 : Cross Product Operation	24
Figure 39 : User view created for the Labs Table	24
Figure 40 : Renaming Operation	25
Figure 41 : Find the Maximum Age from Lab Maintenance Table (Aggregation Functions).....	25
Figure 42 : Use of the keyword 'Like'	26
Figure 43 : Union Operation.....	27
Figure 44 : Intersection Operation.....	27
Figure 45 : Set Difference Operation.....	28

Figure 46 : Division Operation.....	28
Figure 47 : Inner Join between the Labs and Lab Equipment tables with the user view.....	29
Figure 48 : Natural Join between the Labs and Lab Equipment tables with the user view	29
Figure 49 : Left Outer Join between Student and Lab Card tables with user view	30
Figure 50 : Right Outer Join between Student and Lab Card tables with user view	30
Figure 51 : Full Outer Join between Student and Lab Card tables with user view.....	31
Figure 52 : Full Outer Join between Location and Labs tables with user view.....	31
Figure 53 : Outer Union Operation with user view	32
Figure 54 : List of Students Who Scored Above 75	32
Figure 55 : Student's Assessment Marks.....	33
Figure 56 : Students Who Scored Above the Average Mark	33
Figure 57 : Tuned Query 1 and Results Comparing	34
Figure 58 : Tuned Query 2 and Results Comparing	35
Figure 59 : Results for the Original query	35
Figure 60 : Results of the Tuned query.....	36
Figure 61 : Tuned Query 3.....	36
Figure 62 : Tuned query 4 and Results of Tuned query.....	37
Figure 63 : Results of Original query	37
Figure 64 : Tuned query 5 with the Results Comparing.....	38
Figure 65 : Tuned query 6 and Results Comparing	39
Figure 66 : Tuned query 7 with Results Comparing.....	40
Figure 67 : Tuned query 8 with Results Comparing.....	40
Figure 68 : Tuned query 9 with Results Comparing.....	42
Figure 69 : Tuned query 10 with Results Comparing.....	43

INTRODUCTION

Database systems serve as a digitally organized file rack with vast array of features, making storing, tracking, managing and decision-making using information easier. This report dives into the analysis of creating a Laboratory Management System where data integrity and precision are highly crucial. This comprehensive exploration will be discussed under a few main chapters;

- Requirement Analysis
- Conceptual Design
- Implementation
- Transactions
- Database Tuning

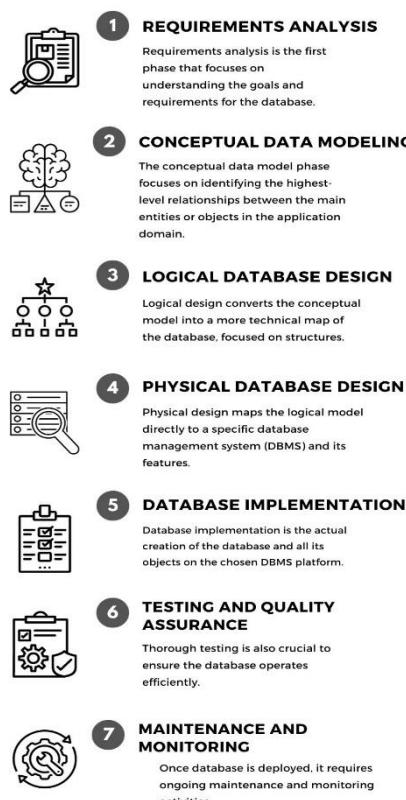


Figure 1 : Main Phases of Databases Designing [1]

The requirement analysis process lays a solid foundation for the mini project. Within Chapter two the graphical representation of our Laboratory Management System is drawn using an ER diagram and the UML Class diagram. Then the database is implemented using MySQL considering to Normalization principles in database schema design. Subsequent chapters, Transactions and Database Tuning will make the Database system for better performances optimizing functionalities.

CHAPTER 1 – REQUIREMENT ANALYSIS

In order to understand the needs of the Database management, analyses of both functional and data requirements have been done, ensuring the effectiveness of the system.

Functional Requirements

- It is necessary to determine the availability status of laboratories, indicating whether they are available for use or already in use.
- In order to manage the maintenance process it is important to know the issues that need to be repaired of each device in each laboratory.
- The location of the each laboratory with the department is required for managing the system as well as for knowledge of students and staff of the university.
- Everyone needs to know which modules are conducted in which laboratories and the information of the relevant module coordinates or the lecturers responsible for those modules.
- The system should display the Assessments details given to the students and marks after evaluations and should be able to calculate the grade using the marks provided by the users.
- System should have the options for the admin to maintain the login process of the staff and the students considering the restrictions on each group.
- Student details including the laboratory details they are enrolled in, such as the lab ID and lab group number, should be visible and manageable.

Data Requirements

To fulfill the data requirements ten entities have been included covering all data as attributes. Those entities and attributes are listed below.

1. LABS

- Lab ID [Primary Key]
- Lab Maintenance Schedule (Lab Maintenance ID,Name, Age, Gender)
- Lab Name

2. LAB EQUIPMENT

- Equipment ID [Primary Key]
- IP Address
- Issues

3. LOCATION

- Location ID [Primary Key]
- Lab Name
- Department

4. STATUS

- Current States
- Network Connectivity
- Date

5. STUDENTS

- Student ID [Primary Key]
- Student Name
- Address (Street No., Street Name, Town)
- Department
- Semester
- User Name
- Password

6. LAB CARD

- Lab Card ID [Primary Key]
- Batch No.

7. MODULES

- Module Code [Primary Key]
- Module Name

8. ASSIGNMENTS

- Assessment ID [Primary Key]
- Name
- Marks
- Grade

9. MODULE COORDINATOR

- Coordinator ID [Primary Key]
- Name
- Department
- Email

10. ADMIN

- Department
- User Name
- Password

CHAPTER 2 – CONCEPTUAL DESIGN

In the Conceptual Design The Entity-Relationship diagram and the UML class diagram were created mentioning all the entities, attributes, relationships, roles names and (min, max) notation. 10 entities were created including 8 strong entities and 2 weak entities.

ER diagram was created using Figma.

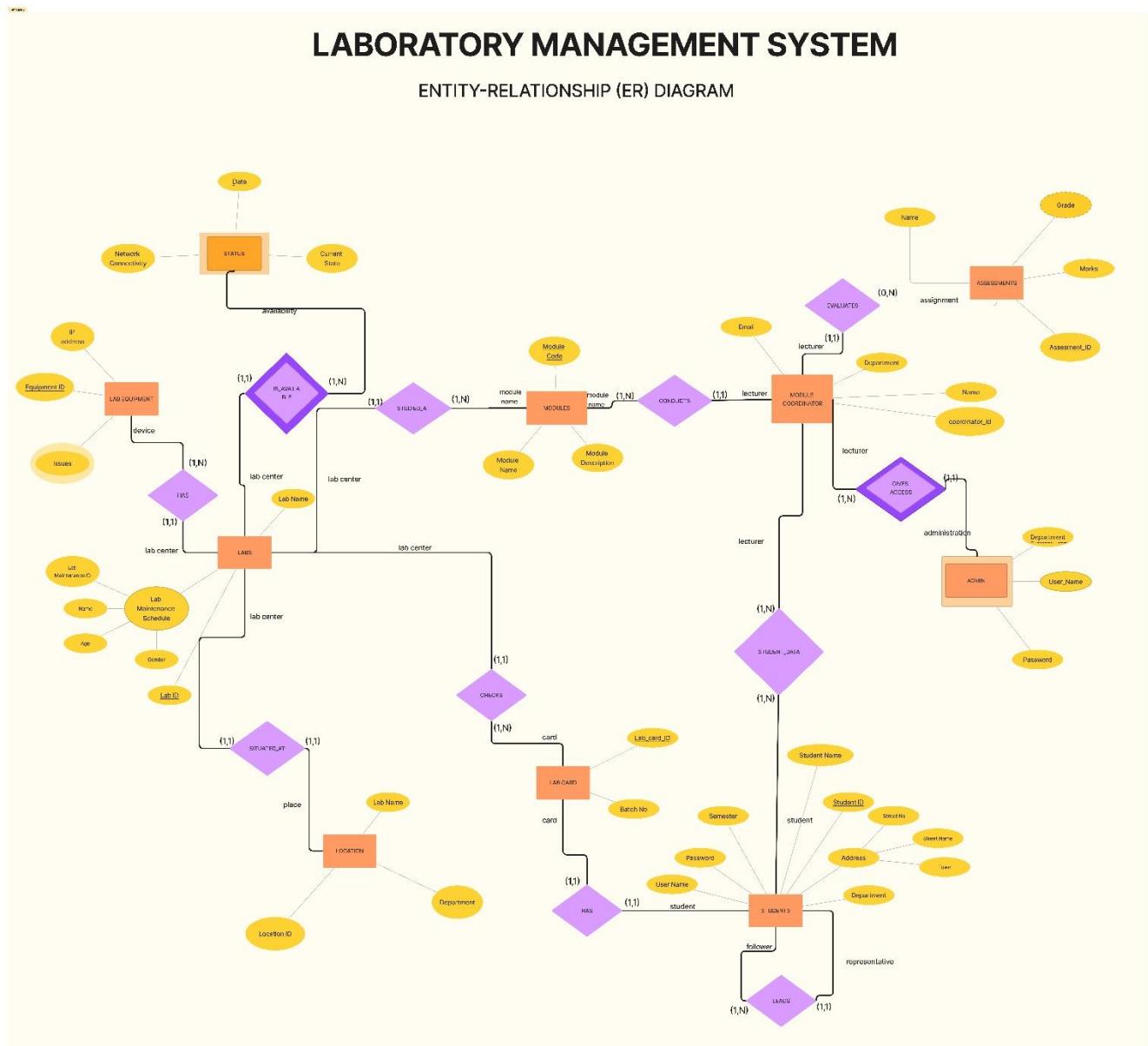


Figure 2 : ER Diagram for the Laboratory Management System

Visual Paradigm was used to create the UML class diagram and it was designed under the normalization principles of second normal form.

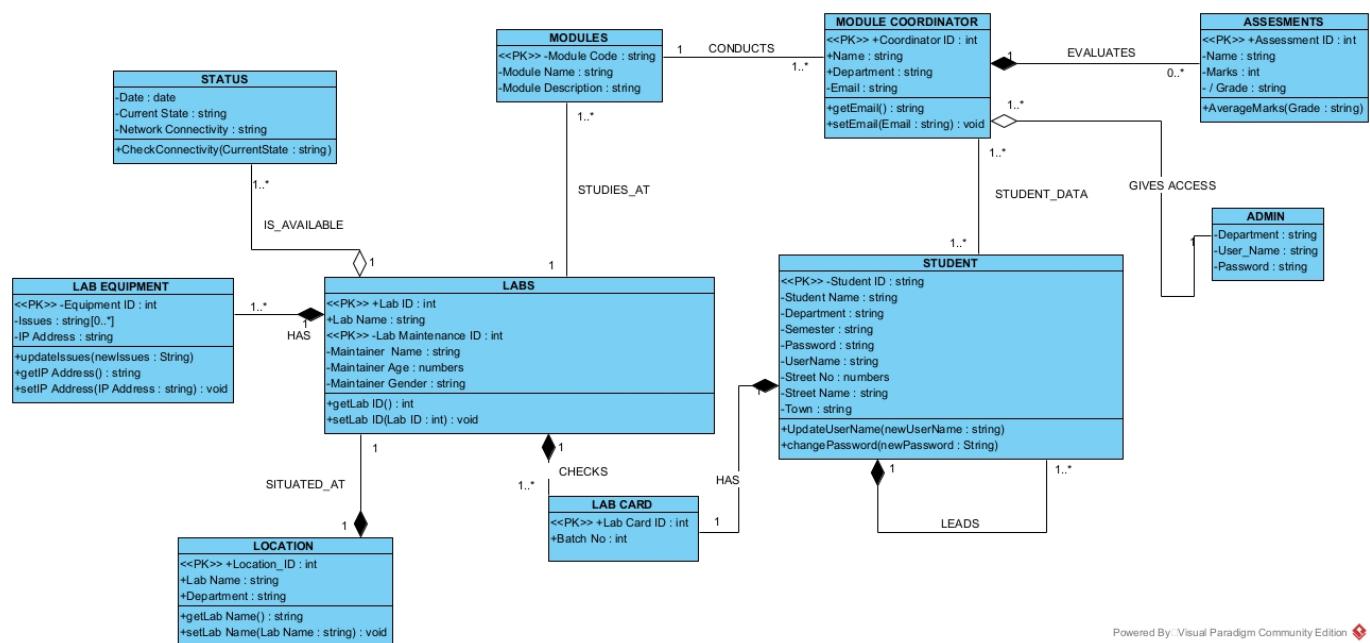
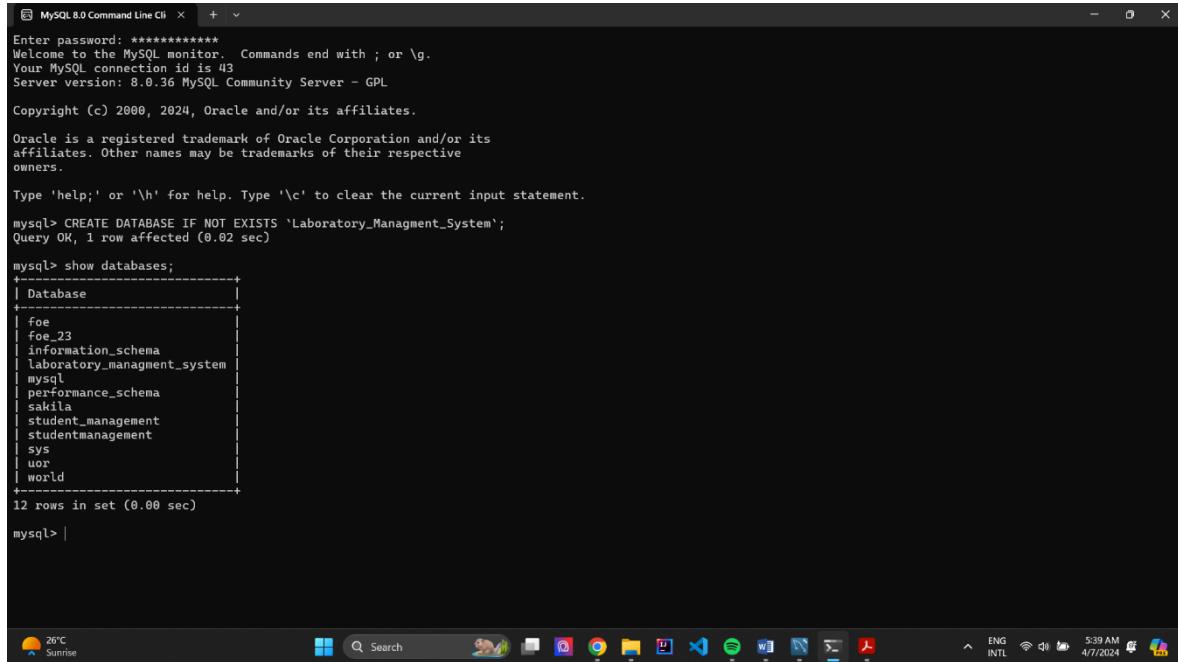


Figure 3 : UML Class Diagram for the Laboratory Management System

Powered By: Visual Paradigm Community Edition

CHAPTER 3 – IMPLEMENTATION

Schema Creation



```
MySQL 8.0 Command Line Cli + 
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 43
Server version: 8.0.36 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

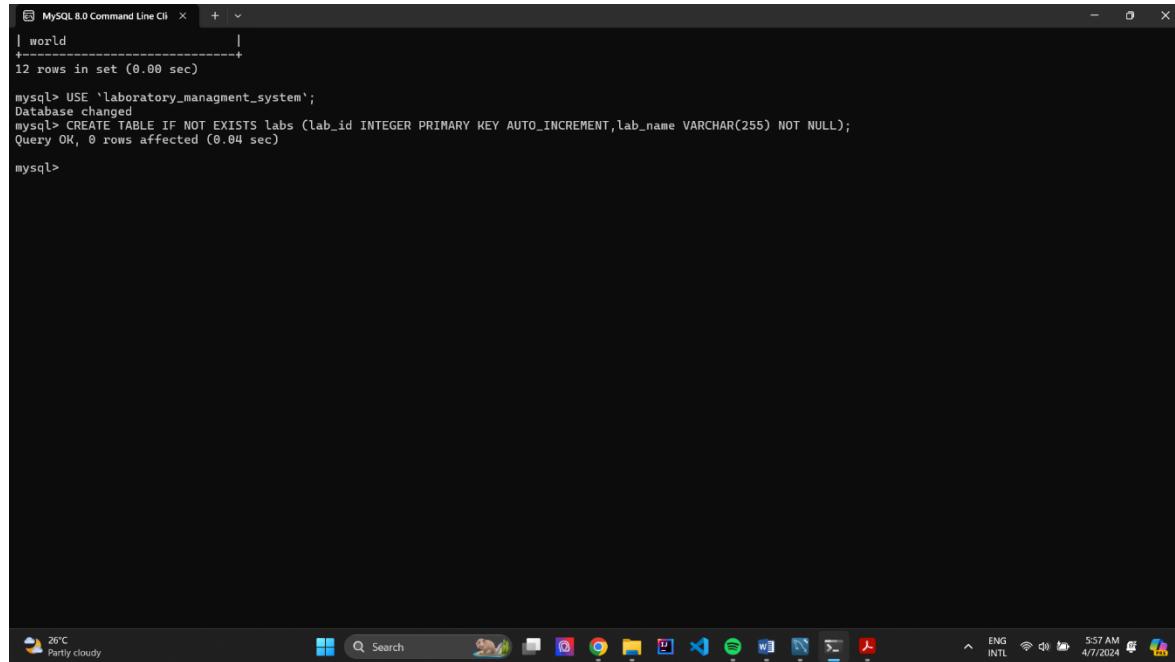
mysql> CREATE DATABASE IF NOT EXISTS 'Laboratory_Management_System';
Query OK, 1 row affected (0.02 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| fne      |
| foe_23   |
| information_schema |
| laboratory_management_system |
| mysql    |
| performance_schema |
| sakila   |
| student_management |
| studentmanagement |
| sys      |
| uor      |
| world    |
+-----+
12 rows in set (0.00 sec)

mysql> |
```

Figure 4 : Creation of Schema Laboratory Management System

Table Definitions

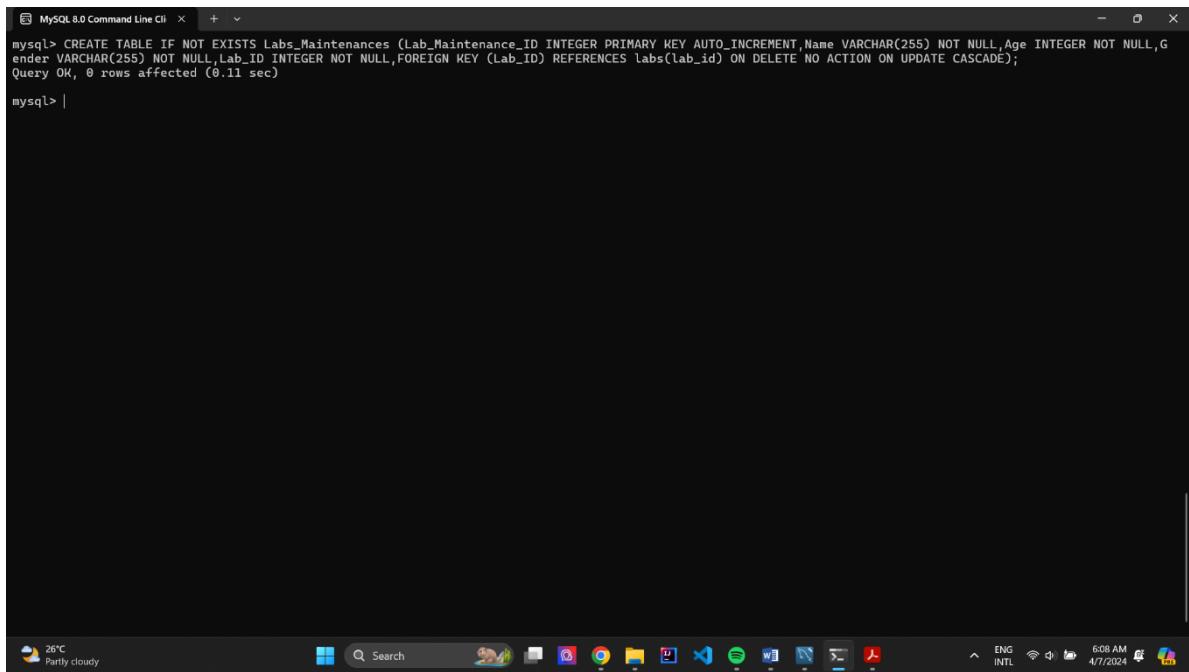


```
MySQL 8.0 Command Line Cli + 
| world      |
+-----+
12 rows in set (0.00 sec)

mysql> USE `laboratory_management_system`;
Database changed
mysql> CREATE TABLE IF NOT EXISTS labs (lab_id INTEGER PRIMARY KEY AUTO_INCREMENT,lab_name VARCHAR(255) NOT NULL);
Query OK, 0 rows affected (0.04 sec)

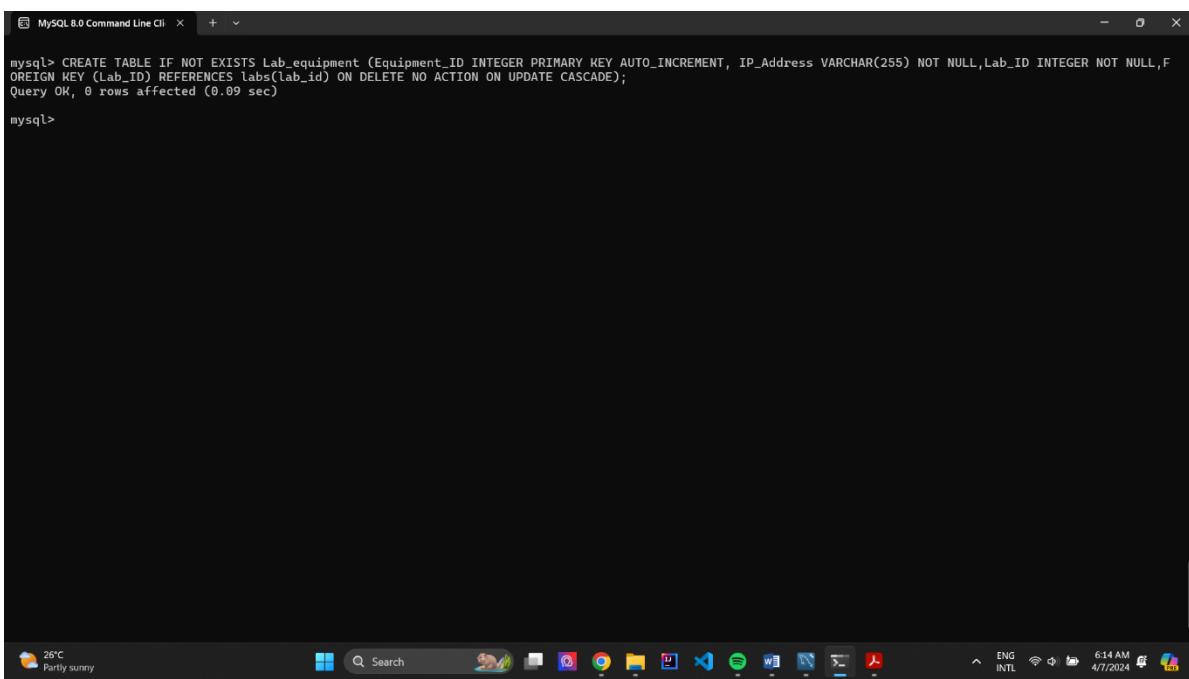
mysql>
```

Figure 5 : Creation of Labs Table



```
MySQL 8.0 Command Line Cli +  
mysql> CREATE TABLE IF NOT EXISTS Labs_Maintenances (Lab_Maintenance_ID INTEGER PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(255) NOT NULL, Age INTEGER NOT NULL, Gender VARCHAR(255) NOT NULL, Lab_ID INTEGER NOT NULL, FOREIGN KEY (Lab_ID) REFERENCES labs(lab_id) ON DELETE NO ACTION ON UPDATE CASCADE);  
Query OK, 0 rows affected (0.11 sec)  
mysql> |
```

Figure 6 : Creation of Lab Maintenance Table



```
MySQL 8.0 Command Line Cli +  
mysql> CREATE TABLE IF NOT EXISTS Lab_equipment (Equipment_ID INTEGER PRIMARY KEY AUTO_INCREMENT, IP_Address VARCHAR(255) NOT NULL, Lab_ID INTEGER NOT NULL, FOREIGN KEY (Lab_ID) REFERENCES labs(lab_id) ON DELETE NO ACTION ON UPDATE CASCADE);  
Query OK, 0 rows affected (0.09 sec)  
mysql>
```

Figure 7 : Creation of Lab Equipment Table

MySQL 8.0 Command Line Cli

```
mysql> CREATE TABLE IF NOT EXISTS Issues (Issues_ID INTEGER PRIMARY KEY AUTO_INCREMENT,Issue VARCHAR(255) NOT NULL,Equipment_ID INTEGER NOT NULL,FOREIGN KEY (Equipment_ID) REFERENCES Lab_equipment(Equipment_ID) ON DELETE NO ACTION ON UPDATE CASCADE;
Query OK, 0 rows affected (0.10 sec)

mysql>
```

26°C Partly sunny

Search

6:33 AM 4/7/2024

Figure 8 : Creation of Issues Table

MySQL 8.0 Command Line Cli

```
mysql> CREATE TABLE IF NOT EXISTS Lab_Status (Date DATE NOT NULL,Current_Status VARCHAR(255) NOT NULL,Network_Connectivity VARCHAR(255) NOT NULL);
Query OK, 0 rows affected (0.06 sec)

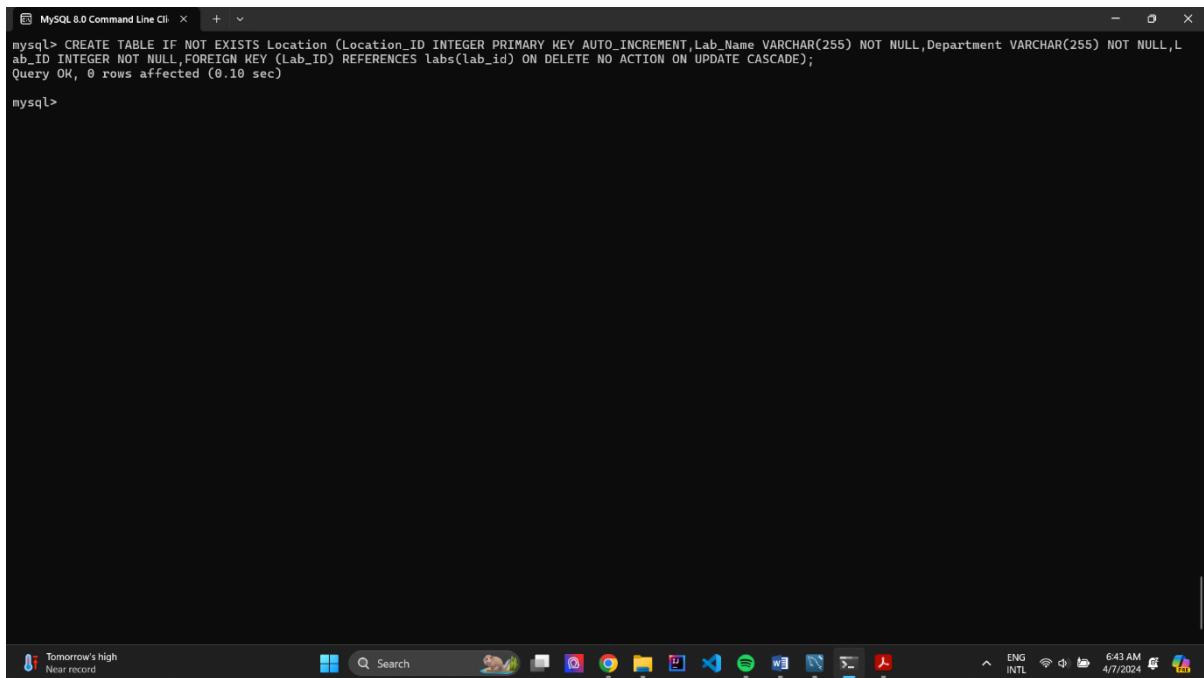
mysql> |
```

26°C Partly sunny

Search

6:37 AM 4/7/2024

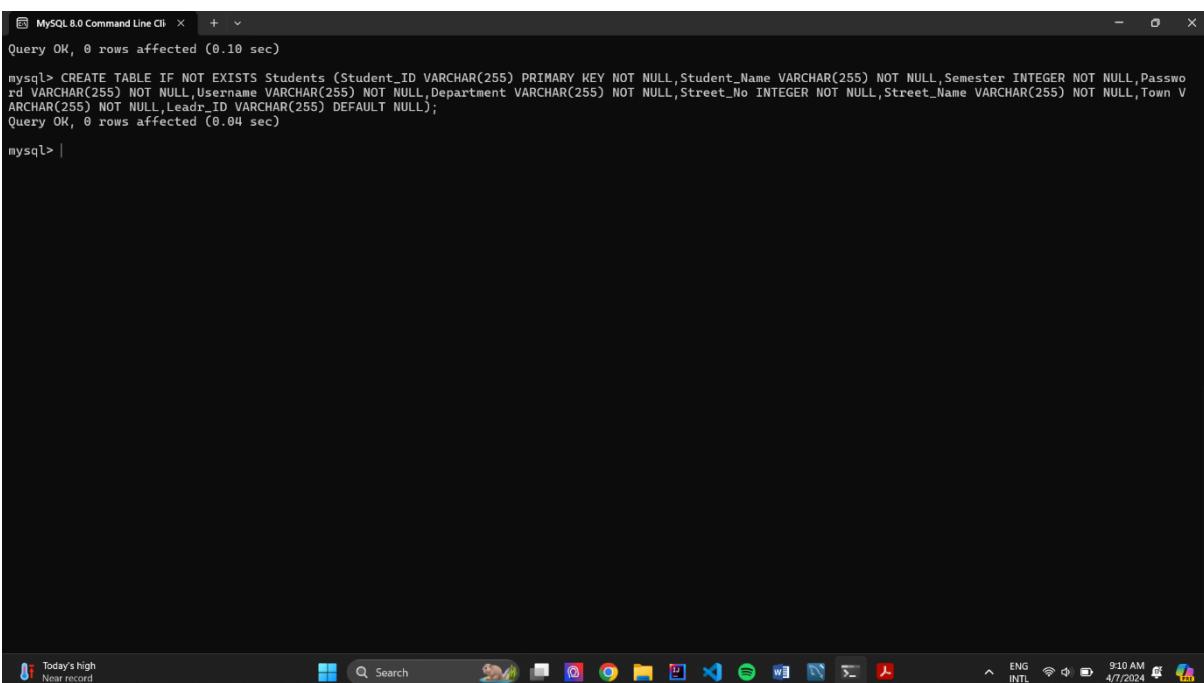
Figure 9 : Creation of Lab Status Table



```
MySQL 8.0 Command Line Cli + v
mysql> CREATE TABLE IF NOT EXISTS Location (Location_ID INTEGER PRIMARY KEY AUTO_INCREMENT,Lab_Name VARCHAR(255) NOT NULL,Department VARCHAR(255) NOT NULL,Lab_ID INTEGER NOT NULL,FOREIGN KEY (Lab_ID) REFERENCES labs(lab_id) ON DELETE NO ACTION ON UPDATE CASCADE);
Query OK, 0 rows affected (0.10 sec)

mysql>
```

Figure 10 : Creation of Location Table



```
MySQL 8.0 Command Line Cli + v
Query OK, 0 rows affected (0.10 sec)

mysql> CREATE TABLE IF NOT EXISTS Students (Student_ID VARCHAR(255) PRIMARY KEY NOT NULL,Student_Name VARCHAR(255) NOT NULL,Semester INTEGER NOT NULL,Passward VARCHAR(255) NOT NULL,Username VARCHAR(255) NOT NULL,Department VARCHAR(255) NOT NULL,Street_No INTEGER NOT NULL,Street_Name VARCHAR(255) NOT NULL,Town VARCHAR(255) NOT NULL,Leadr_ID VARCHAR(255) DEFAULT NULL);
Query OK, 0 rows affected (0.04 sec)

mysql> |
```

Figure 11 : Creation of Students Table

The screenshot shows a Windows desktop environment with a MySQL 8.0 Command Line Client window open. The command entered is:

```
mysql> CREATE TABLE IF NOT EXISTS Lab_Card (Lab_Card_ID INTEGER PRIMARY KEY AUTO_INCREMENT, Student_ID VARCHAR(255) NOT NULL, Batch_No INTEGER NOT NULL, FOREIGN KEY (Student_ID) REFERENCES Students(Student_ID) ON DELETE NO ACTION ON UPDATE CASCADE);
```

The response indicates:

```
Query OK, 0 rows affected (0.09 sec)
```

The MySQL prompt mysql> is visible at the bottom of the window.

Figure 12 : Creation of Lab Card Table

The screenshot shows a Windows desktop environment with a MySQL 8.0 Command Line Client window open. The command entered is:

```
mysql> CREATE TABLE IF NOT EXISTS Modules (Module_Code VARCHAR(255) PRIMARY KEY NOT NULL, Module_Name VARCHAR(255) NOT NULL, Module_Description VARCHAR(255) NOT NULL);
```

The response indicates:

```
Query OK, 0 rows affected (0.02 sec)
```

The MySQL prompt mysql> is visible at the bottom of the window.

Figure 13 : Creation of Modules Table

```
MySQL 8.0 Command Line Cli + X
mysql> CREATE TABLE IF NOT EXISTS module_coordinator (
    mysql> CREATE TABLE IF NOT EXISTS Module_Coordinator (Coordinator_ID VARCHAR(255) PRIMARY KEY NOT NULL,Module_Code VARCHAR(255) NOT NULL,Name VARCHAR(255) NOT NULL,Email VARCHAR(255) NOT NULL,Department VARCHAR(255) NOT NULL,FOREIGN KEY (Module_Code) REFERENCES Modules(Module_Code) ON DELETE NO ACTION ON UPDATE CASCADE);
Query OK, 0 rows affected (0.10 sec)

mysql> |
```

The screenshot shows a Windows desktop environment with a MySQL 8.0 Command Line Client window open. The window title is "MySQL 8.0 Command Line Cli". The command entered is to create a table named "module_coordinator" if it does not already exist. The table structure includes columns for Coordinator_ID (primary key), Module_Code (foreign key referencing "Modules"), Name, Email, Department, and constraints for NOT NULL values and CASCADE updates. The command is completed successfully with 0 rows affected in 0.10 seconds. The system tray at the bottom shows weather (29°C, Partly sunny), network status, battery level, and system time (9:24 AM, 4/7/2024).

Figure 14 : Creation of Module Coordinator Table

```
MySQL 8.0 Command Line Cli + X
mysql> CREATE TABLE IF NOT EXISTS assessments (
    mysql> CREATE TABLE IF NOT EXISTS Assessments (Assessment_ID INTEGER PRIMARY KEY AUTO_INCREMENT,Student_ID VARCHAR(255) NOT NULL,Name VARCHAR(255) NOT NULL,Mark INTEGER NOT NULL DEFAULT 0,Coordinator_ID VARCHAR(255) NOT NULL,FOREIGN KEY (Coordinator_ID) REFERENCES Module_Coordinator(Coordinator_ID) ON DELETE NO ACTION ON UPDATE CASCADE,FOREIGN KEY (Student_ID) REFERENCES Students(Student_ID) ON DELETE NO ACTION ON UPDATE CASCADE);
Query OK, 0 rows affected (0.12 sec)

mysql>
```

The screenshot shows a Windows desktop environment with a MySQL 8.0 Command Line Client window open. The window title is "MySQL 8.0 Command Line Cli". The command entered is to create a table named "assessments" if it does not already exist. The table structure includes columns for Assessment_ID (primary key auto-increment), Student_ID, Name, Mark, and Coordinator_ID (foreign key referencing "Module_Coordinator"). It includes constraints for NOT NULL values and CASCADE updates. The command is completed successfully with 0 rows affected in 0.12 seconds. The system tray at the bottom shows weather (29°C, Partly sunny), network status, battery level, and system time (9:31 AM, 4/7/2024).

Figure 15 : Creation of Assessments Table

```
mysql> CREATE VIEW Assessments_With_Grade AS
-> SELECT Assessment_ID, Student_ID, Name, Mark,
-> CASE
-> WHEN Mark >= 75 THEN 'A'
-> WHEN Mark >= 65 THEN 'B'
-> WHEN Mark >= 50 THEN 'C'
-> WHEN Mark >= 35 THEN 'C-'
-> ELSE 'F'
-> END AS Grade,
-> Coordinator_ID
-> FROM Assessments;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

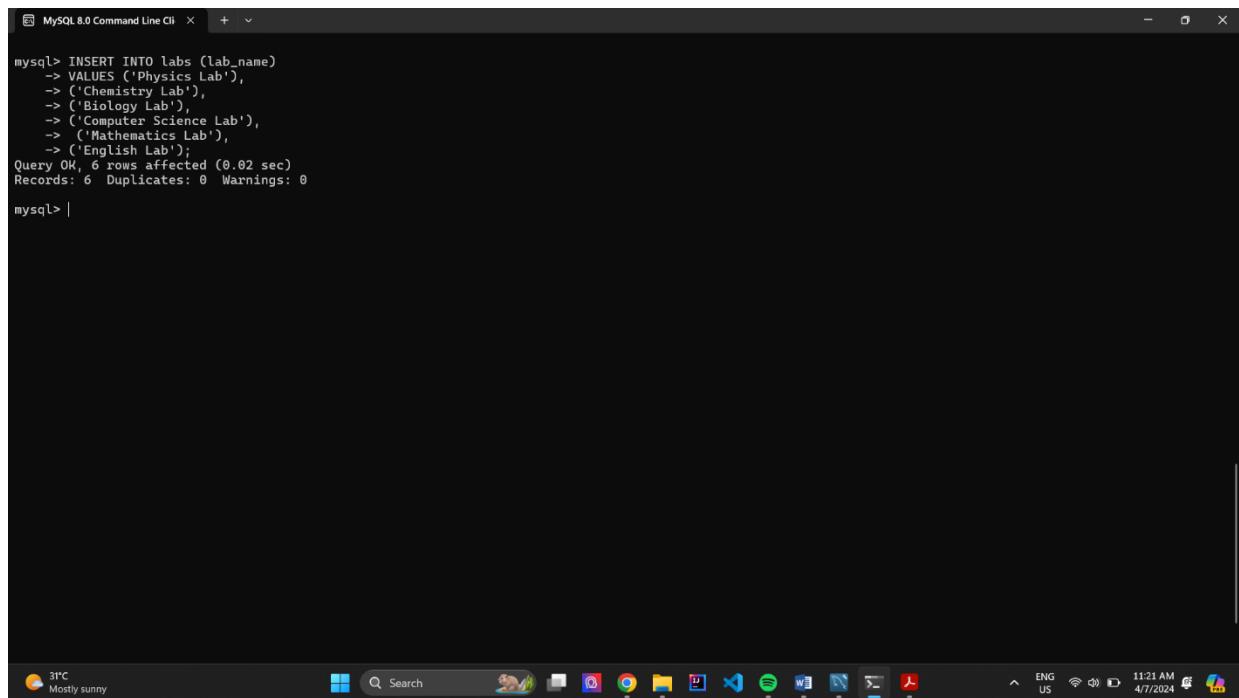
Figure 16 : Definition of the Derived attribute 'Grade'

```
mysql> CREATE TABLE IF NOT EXISTS Admins(User_Name VARCHAR(255) NOT NULL,Password VARCHAR(255) NOT NULL,Department VARCHAR(255) NOT NULL);
Query OK, 0 rows affected (0.03 sec)

mysql> |
```

Figure 17 : Creation of Admins Table

Insertion

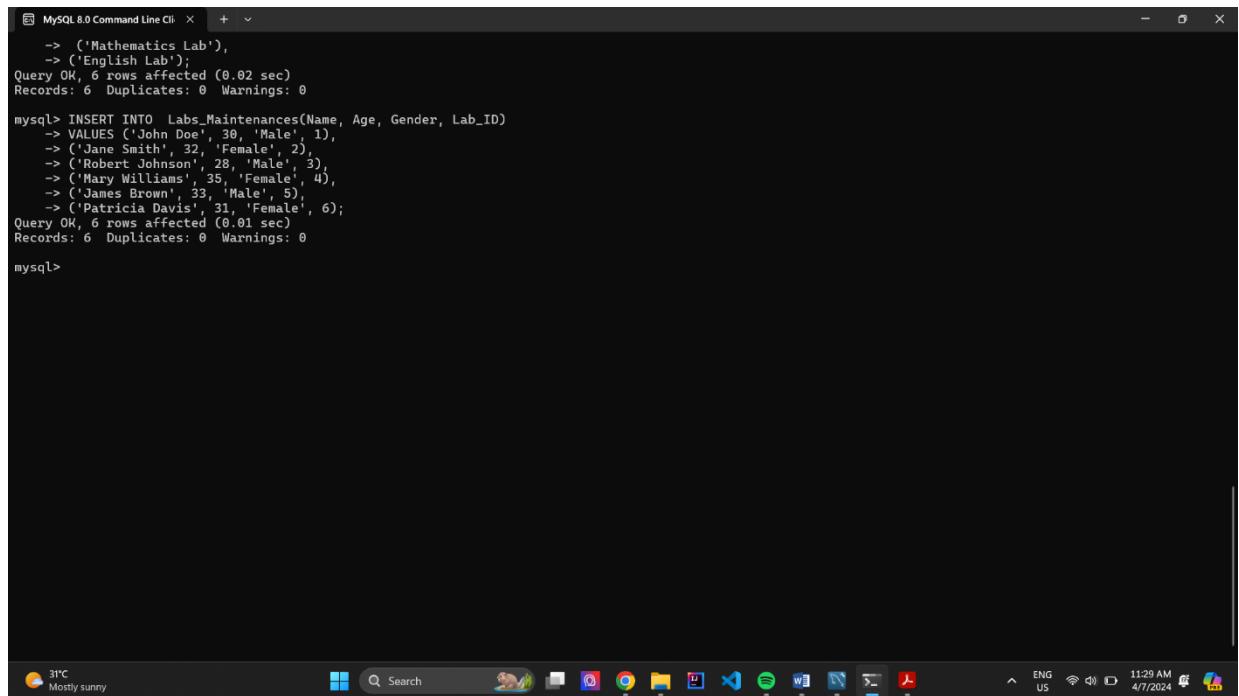


The screenshot shows a Windows desktop environment with a MySQL 8.0 Command Line Cli window open. The command entered is:

```
mysql> INSERT INTO labs (lab_name)
-> VALUES ('Physics Lab'),
-> ('Chemistry Lab'),
-> ('Biology Lab'),
-> ('Computer Science Lab'),
-> ('Mathematics Lab'),
-> ('English Lab');
Query OK, 6 rows affected (0.02 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

The MySQL prompt mysql> is visible at the bottom left of the window.

Figure 18 : Data Insertion to the Labs Table



The screenshot shows a Windows desktop environment with a MySQL 8.0 Command Line Cli window open. The commands entered are:

```
mysql> INSERT INTO `Labs_Maintenances`(`Name`, `Age`, `Gender`, `Lab_ID`)
-> VALUES ('John Doe', 30, 'Male', 1),
-> ('Jane Smith', 32, 'Female', 2),
-> ('Robert Johnson', 28, 'Male', 3),
-> ('Mary Williams', 35, 'Female', 4),
-> ('James Brown', 33, 'Male', 5),
-> ('Patricia Davis', 31, 'Female', 6);
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

The MySQL prompt mysql> is visible at the bottom left of the window.

Figure 19 : Data Insertion to the Lab Maintenance Table

```
mysql> INSERT INTO Lab_equipment (IP_Address, Lab_ID)
-> VALUES ('192.168.1.1', 1),
-> ('192.168.1.2', 2),
-> ('192.168.1.3', 3),
-> ('192.168.1.4', 4),
-> ('192.168.1.5', 5),
-> ('192.168.1.6', 6);
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
mysql>
```

Figure 20 : Data Insertion to the Lab Equipment Table

```
mysql> INSERT INTO Issues (Issue, Equipment_ID)
-> VALUES ('Issue 1', 2),
-> ('Issue 2', 2),
-> ('Issue 3', 3),
-> ('Issue 4', 4),
-> ('Issue 5', 5),
-> ('Issue 6', 6);
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
mysql> |
```

Figure 21 : Data Insertion to the Issues Table

```
mysql> INSERT INTO Lab_Status (Date, Current_Status, Network_Connectivity)
-> VALUES ('2022-01-01', 'Operational', 'Connected'),
-> ('2022-02-01', 'Operational', 'Disconnected'),
-> ('2022-03-01', 'Under Maintenance', 'Connected'),
-> ('2022-04-01', 'Operational', 'Connected'),
-> ('2022-05-01', 'Under Maintenance', 'Disconnected'),
-> ('2022-06-01', 'Operational', 'Connected');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
mysql> |
```

Figure 22 : Data Insertion to the Lab Status Table

```
mysql> INSERT INTO Location (Lab_Name, Department, Lab_ID)
-> VALUES ('Physics Lab', 'Physics Department', 1),
-> ('Chemistry Lab', 'Chemistry Department', 2),
-> ('Biology Lab', 'Biology Department', 3),
-> ('Computer Science Lab', 'Computer Science Department', 4),
-> ('Mathematics Lab', 'Mathematics Department', 5),
-> ('English Lab', 'English Department', 6);
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
mysql> |
```

Figure 23 : Data Insertion to the Location Table

The screenshot shows a Windows desktop environment with a MySQL 8.0 Command Line Client window open. The command entered is:

```
mysql> INSERT INTO Students (Student_ID, Student_Name, Semester, Password, Username, Department, Street_No, Street_Name, Town)
-> VALUES ('S1', 'Student 1', 1, 'password1', 'username1', 'Computer Science', 123, 'Street 1', 'Town 1'),
-> ('S2', 'Student 2', 2, 'password2', 'username2', 'Computer Science', 456, 'Street 2', 'Town 2'),
-> ('S3', 'Student 3', 1, 'password3', 'username3', 'Computer Science', 789, 'Street 3', 'Town 3'),
-> ('S4', 'Student 4', 2, 'password4', 'username4', 'Mathematics', 321, 'Street 4', 'Town 4'),
-> ('S5', 'Student 5', 1, 'password5', 'username5', 'English', 654, 'Street 5', 'Town 5'),
-> ('S6', 'Student 6', 2, 'password6', 'username6', 'Physics', 987, 'Street 6', 'Town 6');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

mysql> |

Figure 24 : Data Insertion to the Students Table

The screenshot shows a Windows desktop environment with a MySQL 8.0 Command Line Client window open. The command entered is:

```
mysql> INSERT INTO Lab_Card (Student_ID, Batch_No)
-> VALUES ('S1', 2021),
-> ('S2', 2021),
-> ('S3', 2022),
-> ('S4', 2021),
-> ('S5', 2022),
-> ('S6', 2021);
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

mysql> |

Figure 25 : Data Insertion to the Lab Card Table

The screenshot shows a Windows desktop environment with a MySQL 8.0 Command Line Client window open. The window title is "MySQL 8.0 Command Line Cli". The command entered is:

```
mysql> INSERT INTO Modules (Module_Code, Module_Name, Module_Description)
-> VALUES ('CS101', 'Computer Science Basics', 'Introduction to Computer Science'),
-> ('CS102', 'Advanced Computer Science', 'Advanced topics in Computer Science'),
-> ('MA101', 'Mathematics Basics', 'Introduction to Mathematics'),
-> ('EN101', 'English Basics', 'Introduction to English'),
-> ('PH101', 'Physics Basics', 'Introduction to Physics'),
-> ('CH101', 'Chemistry Basics', 'Introduction to Chemistry');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

mysql>

Figure 26 : Data Insertion to the Modules Table

The screenshot shows a Windows desktop environment with a MySQL 8.0 Command Line Client window open. The window title is "MySQL 8.0 Command Line Cli". The command entered is:

```
mysql> INSERT INTO Module_Coordinator (Coordinator_ID, Module_Code, Name, Email, Department)
-> VALUES ('C1', 'CS101', 'Dr.Emily Johnson', 'emily.johnson@google.com', 'Computer Science'),
-> ('C2', 'CS102', 'Michael Brown', 'michael.brown@yahoo.com', 'Computer Science'),
-> ('C3', 'MA101', 'Robert Johnson', 'robertjohnson@example.com', 'Mathematics'),
-> ('C4', 'EN101', 'Sarah Lee', 'sarah.lee@gmail.com', 'English'),
-> ('C5', 'PH101', 'David Wang', 'david.wang@hotmail.com', 'Physics'),
-> ('C6', 'CH101', 'Ms.Lisa Taylor', 'lisa.taylor@yahoo.com', 'Chemistry');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

mysql>

Figure 27 : Data Insertion to the Module Coordinator Table

The screenshot shows a Windows desktop environment with a MySQL 8.0 Command Line Client window open. The command entered is:

```
mysql> INSERT INTO Assessments (Student_ID, Name, Mark, Coordinator_ID)
-> VALUES ('S1', 'Assessment 1', 85, 'C1'),
-> ('S2', 'Assessment 2', 90, 'C2'),
-> ('S3', 'Assessment 3', 95, 'C3'),
-> ('S4', 'Assessment 4', 80, 'C4'),
-> ('S5', 'Assessment 5', 75, 'C5'),
-> ('S6', 'Assessment 6', 70, 'C6');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

The MySQL prompt mysql> is visible at the bottom of the window. The system tray at the bottom of the screen shows the date and time as 4/7/2024 12:19 PM.

Figure 28 : Data Insertion to the Assessments Table

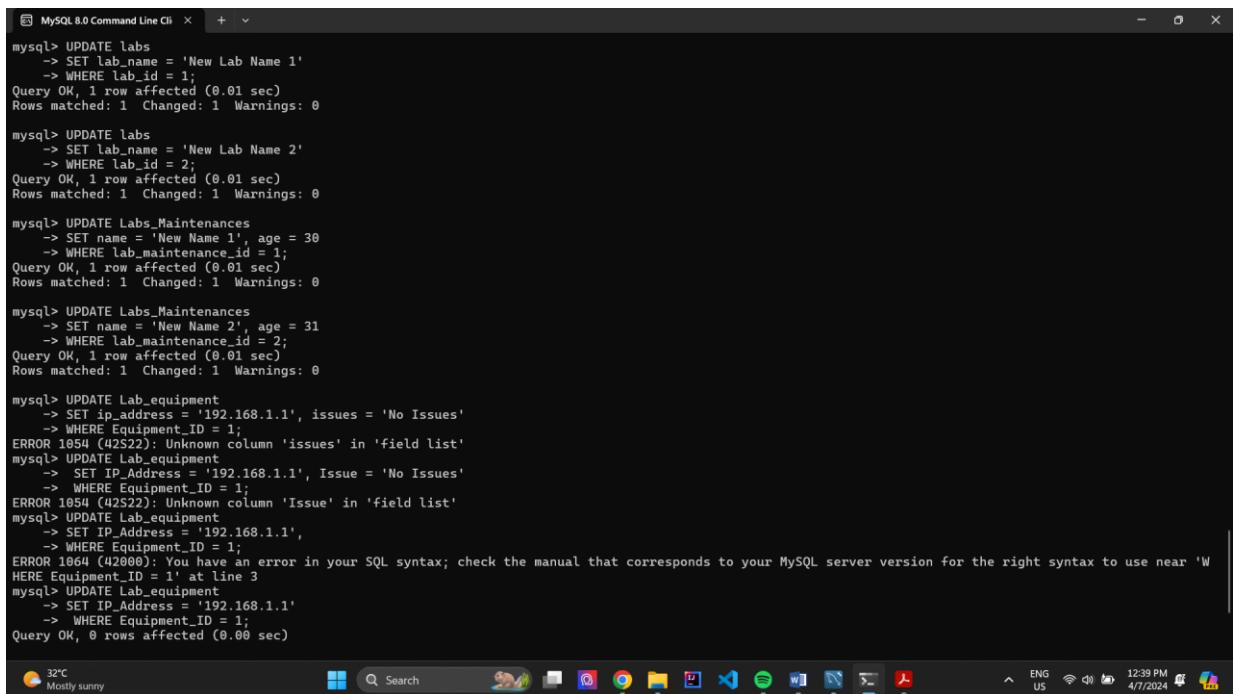
The screenshot shows a Windows desktop environment with a MySQL 8.0 Command Line Client window open. The command entered is:

```
mysql> INSERT INTO Admins (User_Name, Password, Department)
-> VALUES ('admin1', 'password1', 'Computer Science'),
-> ('admin2', 'password2', 'Mathematics'),
-> ('admin3', 'password3', 'Physics'),
-> ('admin4', 'password4', 'Chemistry'),
-> ('admin5', 'password5', 'Biology'),
-> ('admin6', 'password6', 'English');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

The MySQL prompt mysql> is visible at the bottom of the window. The system tray at the bottom of the screen shows the date and time as 4/7/2024 12:22 PM.

Figure 29 : Data Insertion to the Admins Table

Update Operation



```
mysql> UPDATE labs
    -> SET lab_name = 'New Lab Name 1'
    -> WHERE lab_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE labs
    -> SET lab_name = 'New Lab Name 2'
    -> WHERE lab_id = 2;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

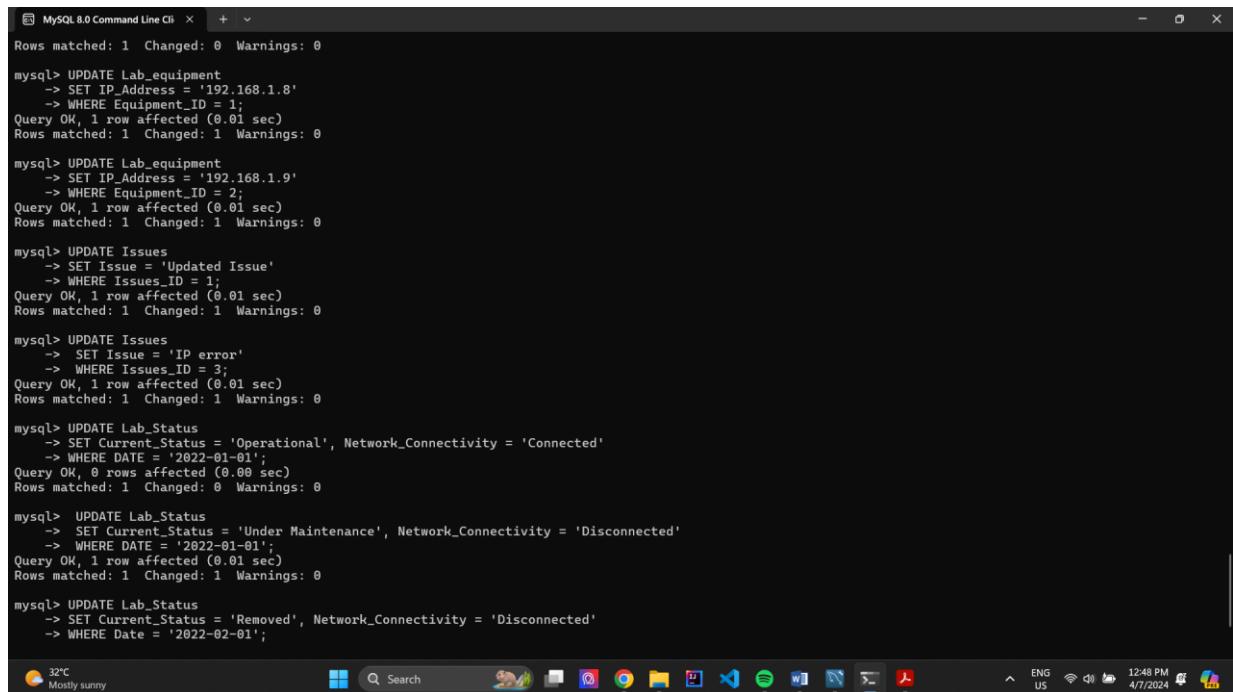
mysql> UPDATE Labs_Maintenances
    -> SET name = 'New Name 1', age = 30
    -> WHERE lab_maintenance_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Labs_Maintenances
    -> SET name = 'New Name 2', age = 31
    -> WHERE lab_maintenance_id = 2;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Lab_equipment
    -> SET ip_address = '192.168.1.1', issues = 'No Issues'
    -> WHERE Equipment_ID = 1;
ERROR 1054 (42S22): Unknown column 'issues' in 'field list'
mysql> UPDATE Lab_equipment
    -> SET IP_Address = '192.168.1.1', Issue = 'No Issues'
    -> WHERE Equipment_ID = 1;
ERROR 1054 (42S22): Unknown column 'Issue' in 'field list'
mysql> UPDATE Lab_equipment
    -> SET IP_Address = '192.168.1.1',
    -> WHERE Equipment_ID = 1;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'W
HERE Equipment_ID = 1' at line 3
mysql> UPDATE Lab_equipment
    -> SET IP_Address = '192.168.1.1'
    -> WHERE Equipment_ID = 1;
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE Lab_equipment
    -> WHERE Equipment_ID = 1;
```

Figure 30 : Data Updates in Labs and Lab Maintenance Tables



```
Rows matched: 1 Changed: 0 Warnings: 0

mysql> UPDATE Lab_equipment
    -> SET IP_Address = '192.168.1.8'
    -> WHERE Equipment_ID = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Lab_equipment
    -> SET IP_Address = '192.168.1.9'
    -> WHERE Equipment_ID = 2;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Issues
    -> SET Issue = 'Updated Issue'
    -> WHERE Issues_ID = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Issues
    -> SET Issue = 'IP error'
    -> WHERE Issues_ID = 3;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Lab_Status
    -> SET Current_Status = 'Operational', Network_Connectivity = 'Connected'
    -> WHERE DATE = '2022-01-01';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1 Changed: 0 Warnings: 0

mysql> UPDATE Lab_Status
    -> SET Current_Status = 'Under Maintenance', Network_Connectivity = 'Disconnected'
    -> WHERE DATE = '2022-01-01';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Lab_Status
    -> SET Current_Status = 'Removed', Network_Connectivity = 'Disconnected'
    -> WHERE Date = '2022-02-01';
```

Figure 31 : Data Updates in Lab Equipment and Issues Tables

```
MySQL 8.0 Command Line Cli + ^ X
Rows matched: 1 Changed: 1 Warnings: 0
mysql> UPDATE Lab_Status
-> SET Current_Status = 'Operational', Network_Connectivity = 'Connected'
-> WHERE DATE = '2022-01-01';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1 Changed: 0 Warnings: 0

mysql> UPDATE Lab_Status
-> SET Current_Status = 'Under Maintenance', Network_Connectivity = 'Disconnected'
-> WHERE DATE = '2022-01-01';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Lab_Status
-> SET Current_Status = 'Removed', Network_Connectivity = 'Disconnected'
-> WHERE Date = '2022-02-01';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Location
-> SET Lab_Name = 'New Lab Name 1', department = 'New Department 1'
-> WHERE Location_ID = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Location
-> SET Lab_Name = 'New Lab Name 2', department = 'New Department 2'
-> WHERE Location_ID = 2;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql>
```

Figure 32 : Data Updates in Lab Status and Location Tables

```
MySQL 8.0 Command Line Cli + ^ X
Rows matched: 1 Changed: 1 Warnings: 0
mysql> UPDATE Students
-> SET Student_Name = 'New Student Name 1', Semester = 2
-> WHERE Student_ID = 'S1';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Students
-> SET Student_Name = 'New Student Name 2', Semester = 3
-> WHERE Student_ID = 'S2';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Lab_Card
-> SET Batch_No = 2022
-> WHERE Lab_Card_ID = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Lab_Card
-> SET Batch_No = 2023
-> WHERE Lab_Card_ID = 2;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Modules
-> SET Module_Name = 'New Module Name 1', Module_Description = 'New Description 1'
-> WHERE Module_Code = 'CS101';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Modules
-> SET Module_Name = 'New Module Name 2', Module_Description = 'New Description 2'
-> WHERE Module_Code = 'CS102';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> |
```

Figure 33 : Data Updates in Students, Lab Card and Modules Tables

```
MySQL 8.0 Command Line Cli × + ▾
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Module_Coordinator
    -> SET Name = 'Mr.David Wang'
    -> WHERE Coordinator_ID = 'C5';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Module_Coordinator
    -> SET Name = 'Mr.David Young'
    -> WHERE Coordinator_ID = 'C5';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE Assessments
    -> SET name = 'New Assessment Name 1', mark = 90
    -> WHERE assessment_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE assessments
    -> SET name = 'New Assessment Name 2', mark = 95
    -> WHERE assessment_id = 2;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE admins
    -> SET password = 'newpassword1', department = 'New Department 1'
    -> WHERE user_name = 'admin1';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE admins
    -> SET password = 'newpassword2', department = 'New Department 2'
    -> WHERE user_name = 'admin2';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

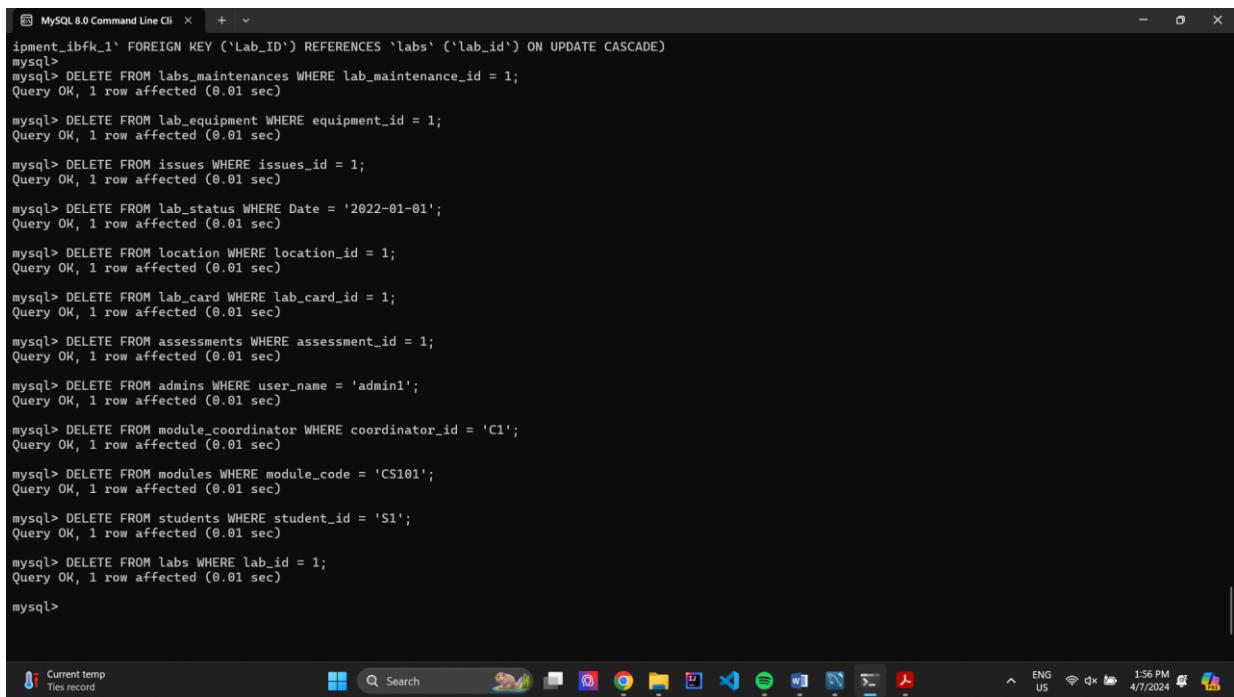
mysql>



```

Figure 34 : Data Updates in Module Coordinator, Assessments and Admins Tables

Deletion



```
MySQL 8.0 Command Line Cli  +  ~
ipment_ibfk_1` FOREIGN KEY (`Lab_ID`) REFERENCES `labs` (`lab_id`) ON UPDATE CASCADE)
mysql> DELETE FROM labs_maintenances WHERE lab_maintenance_id = 1;
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM lab_equipment WHERE equipment_id = 1;
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM issues WHERE issues_id = 1;
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM lab_status WHERE Date = '2022-01-01';
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM location WHERE location_id = 1;
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM lab_card WHERE lab_card_id = 1;
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM assessments WHERE assessment_id = 1;
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM admins WHERE user_name = 'admin1';
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM module_coordinator WHERE coordinator_id = 'C1';
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM modules WHERE module_code = 'CS101';
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM students WHERE student_id = 'S1';
Query OK, 1 row affected (0.01 sec)

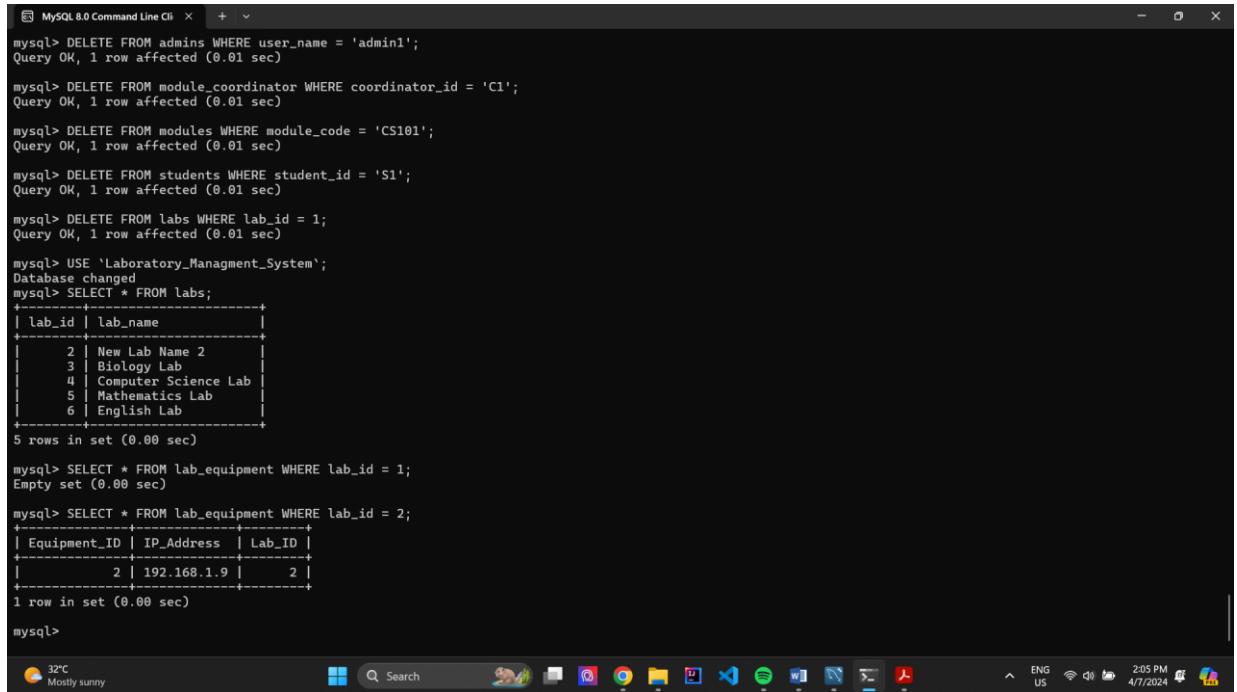
mysql> DELETE FROM labs WHERE lab_id = 1;
Query OK, 1 row affected (0.01 sec)

mysql>
```

Figure 35 : Deleting data from each table

CHAPTER 4 – TRANSACTION

Simple Queries



```
mysql> DELETE FROM admins WHERE user_name = 'admin1';
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM module_coordinator WHERE coordinator_id = 'C1';
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM modules WHERE module_code = 'CS101';
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM students WHERE student_id = 'S1';
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM labs WHERE lab_id = 1;
Query OK, 1 row affected (0.01 sec)

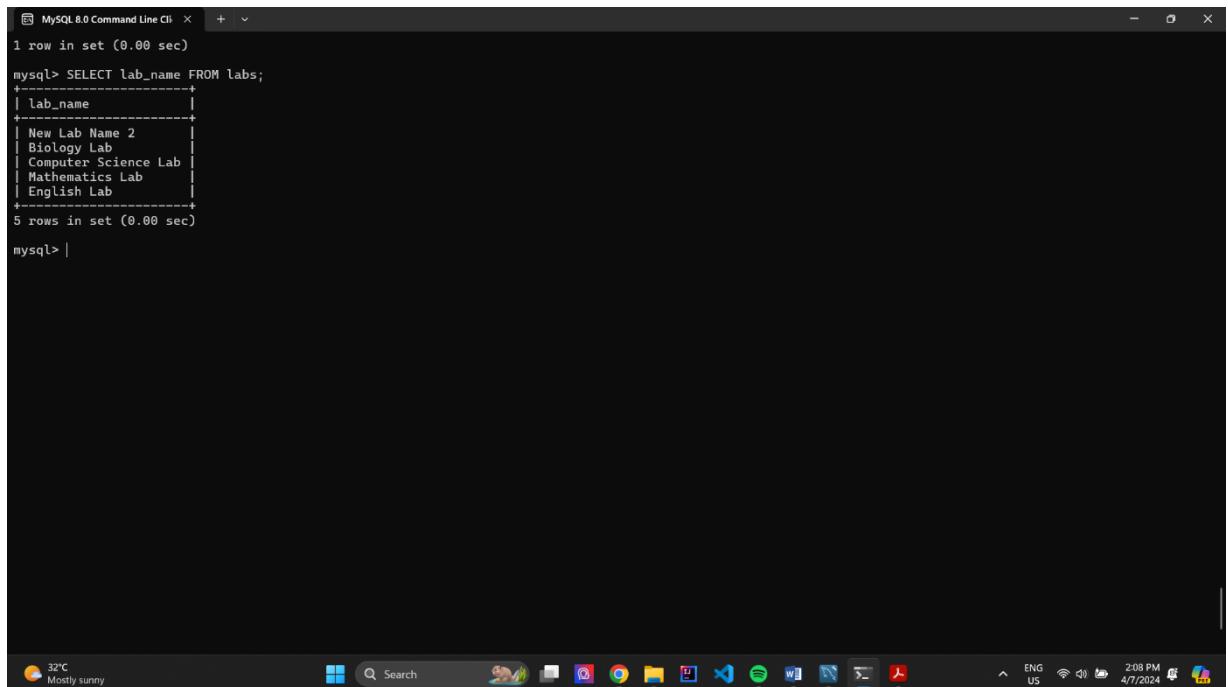
mysql> USE 'Laboratory_Management_System';
Database changed
mysql> SELECT * FROM labs;
+-----+-----+
| lab_id | lab_name |
+-----+-----+
| 2     | New Lab Name 2
| 3     | Biology Lab
| 4     | Computer Science Lab
| 5     | Mathematics Lab
| 6     | English Lab
+-----+-----+
5 rows in set (0.00 sec)

mysql> SELECT * FROM lab_equipment WHERE lab_id = 1;
Empty set (0.00 sec)

mysql> SELECT * FROM lab_equipment WHERE lab_id = 2;
+-----+-----+-----+
| Equipment_ID | IP_Address | Lab_ID |
+-----+-----+-----+
| 2             | 192.168.1.9 | 2       |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 37 : Select Operation



```
1 row in set (0.00 sec)

mysql> SELECT lab_name FROM labs;
+-----+
| lab_name |
+-----+
| New Lab Name 2
| Biology Lab
| Computer Science Lab
| Mathematics Lab
| English Lab
+-----+
5 rows in set (0.00 sec)

mysql> |
```

Figure 36 : Project Operation

MySQL 8.0 Command Line Cli

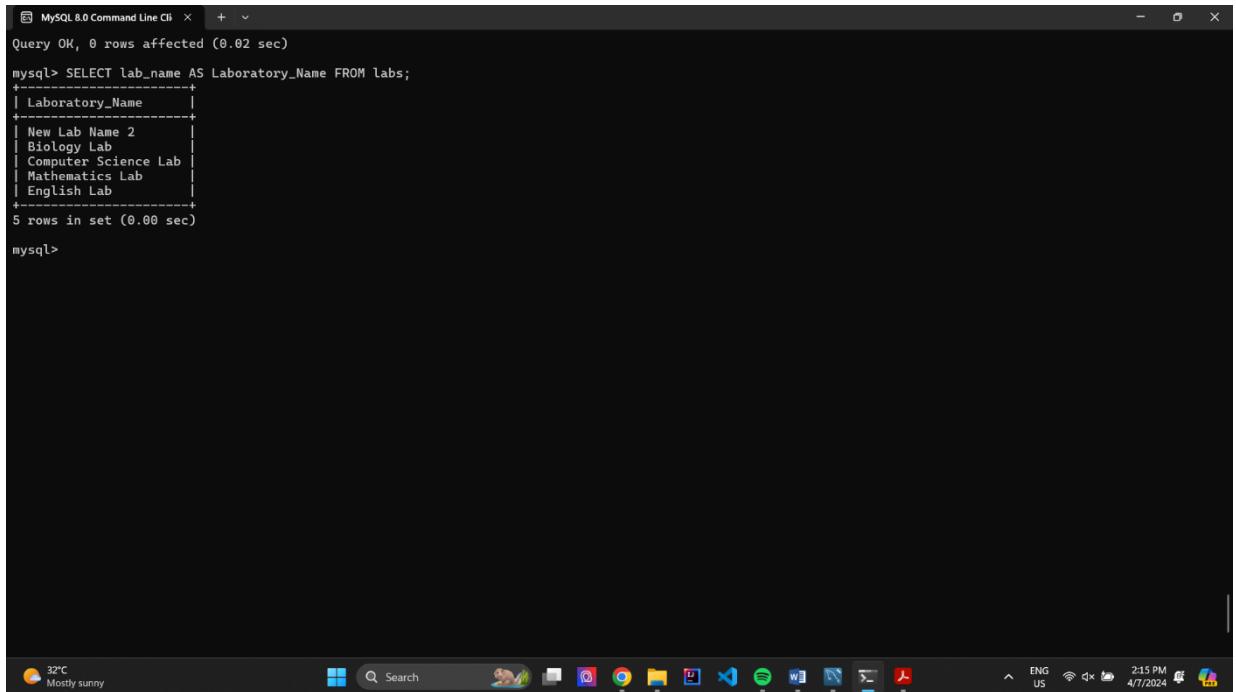
```
+-----+  
5 rows in set (0.00 sec)  
  
mysql> SELECT * FROM labs CROSS JOIN lab_equipment;  
+-----+  
| lab_id | lab_name           | Equipment_ID | IP_Address | Lab_ID |  
+-----+  
| 6      | English Lab          | 2            | 192.168.1.9 | 2     |  
| 5      | Mathematics Lab       | 2            | 192.168.1.9 | 2     |  
| 4      | Computer Science Lab  | 2            | 192.168.1.9 | 2     |  
| 3      | Biology Lab           | 2            | 192.168.1.9 | 2     |  
| 2      | New Lab Name 2         | 2            | 192.168.1.9 | 2     |  
| 6      | English Lab          | 3            | 192.168.1.3 | 3     |  
| 5      | Mathematics Lab       | 3            | 192.168.1.3 | 3     |  
| 4      | Computer Science Lab  | 3            | 192.168.1.3 | 3     |  
| 3      | Biology Lab           | 3            | 192.168.1.3 | 3     |  
| 2      | New Lab Name 2         | 3            | 192.168.1.3 | 3     |  
| 6      | English Lab          | 4            | 192.168.1.4 | 4     |  
| 5      | Mathematics Lab       | 4            | 192.168.1.4 | 4     |  
| 4      | Computer Science Lab  | 4            | 192.168.1.4 | 4     |  
| 3      | Biology Lab           | 4            | 192.168.1.4 | 4     |  
| 2      | New Lab Name 2         | 4            | 192.168.1.4 | 4     |  
| 6      | English Lab          | 5            | 192.168.1.5 | 5     |  
| 5      | Mathematics Lab       | 5            | 192.168.1.5 | 5     |  
| 4      | Computer Science Lab  | 5            | 192.168.1.5 | 5     |  
| 3      | Biology Lab           | 5            | 192.168.1.5 | 5     |  
| 2      | New Lab Name 2         | 5            | 192.168.1.5 | 5     |  
| 6      | English Lab          | 6            | 192.168.1.6 | 6     |  
| 5      | Mathematics Lab       | 6            | 192.168.1.6 | 6     |  
| 4      | Computer Science Lab  | 6            | 192.168.1.6 | 6     |  
| 3      | Biology Lab           | 6            | 192.168.1.6 | 6     |  
| 2      | New Lab Name 2         | 6            | 192.168.1.6 | 6     |  
+-----+  
25 rows in set (0.00 sec)  
  
mysql>
```

Figure 38 : Cross Product Operation

MySQL 8.0 Command Line Cli

```
+-----+  
3 rows in set (0.00 sec)  
  
mysql> CREATE VIEW lab_view AS SELECT lab_name FROM labs;  
Query OK, 0 rows affected (0.02 sec)  
  
mysql>
```

Figure 39 : User view created for the Labs Table



MySQL 8.0 Command Line Cli

```
Query OK, 0 rows affected (0.02 sec)

mysql> SELECT lab_name AS Laboratory_Name FROM labs;
+-----+
| Laboratory_Name |
+-----+
| New Lab Name 2 |
| Biology Lab    |
| Computer Science Lab |
| Mathematics Lab |
| English Lab    |
+-----+
5 rows in set (0.00 sec)

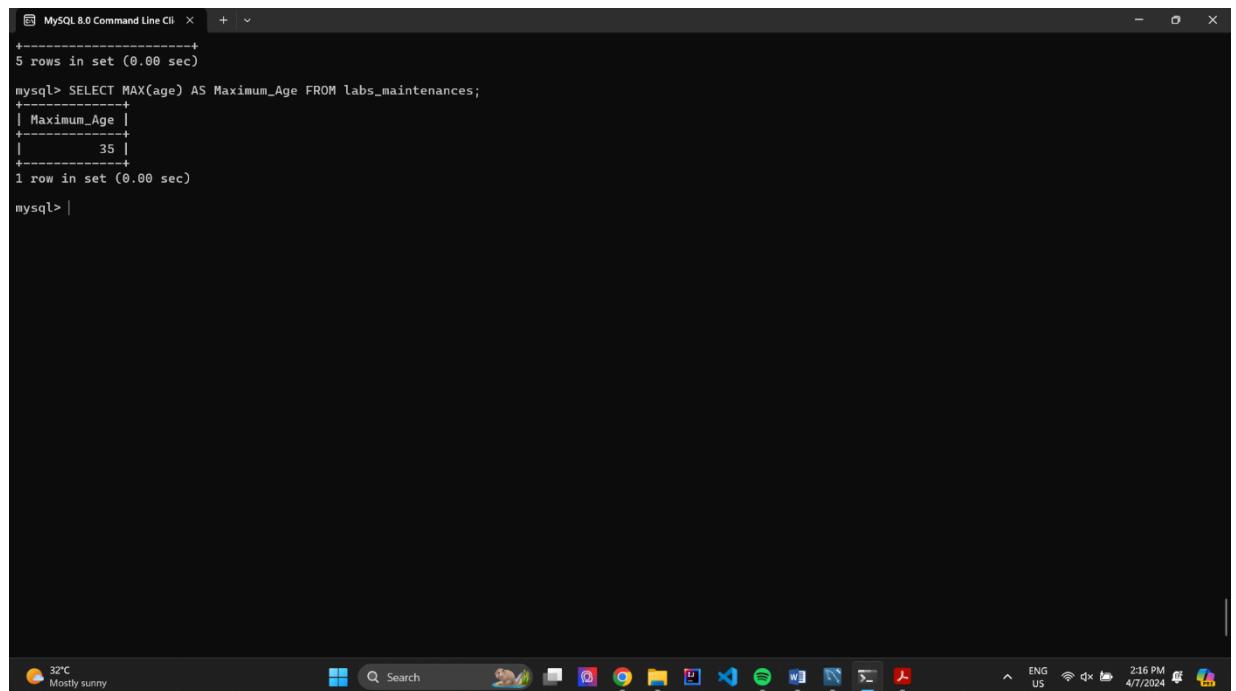
mysql>
```

32°C Mostly sunny

Search

215 PM 4/7/2024

Figure 40 : Renaming Operation



MySQL 8.0 Command Line Cli

```
+-----+
5 rows in set (0.00 sec)

mysql> SELECT MAX(age) AS Maximum_Age FROM labs_maintenances;
+-----+
| Maximum_Age |
+-----+
|      35     |
+-----+
1 row in set (0.00 sec)

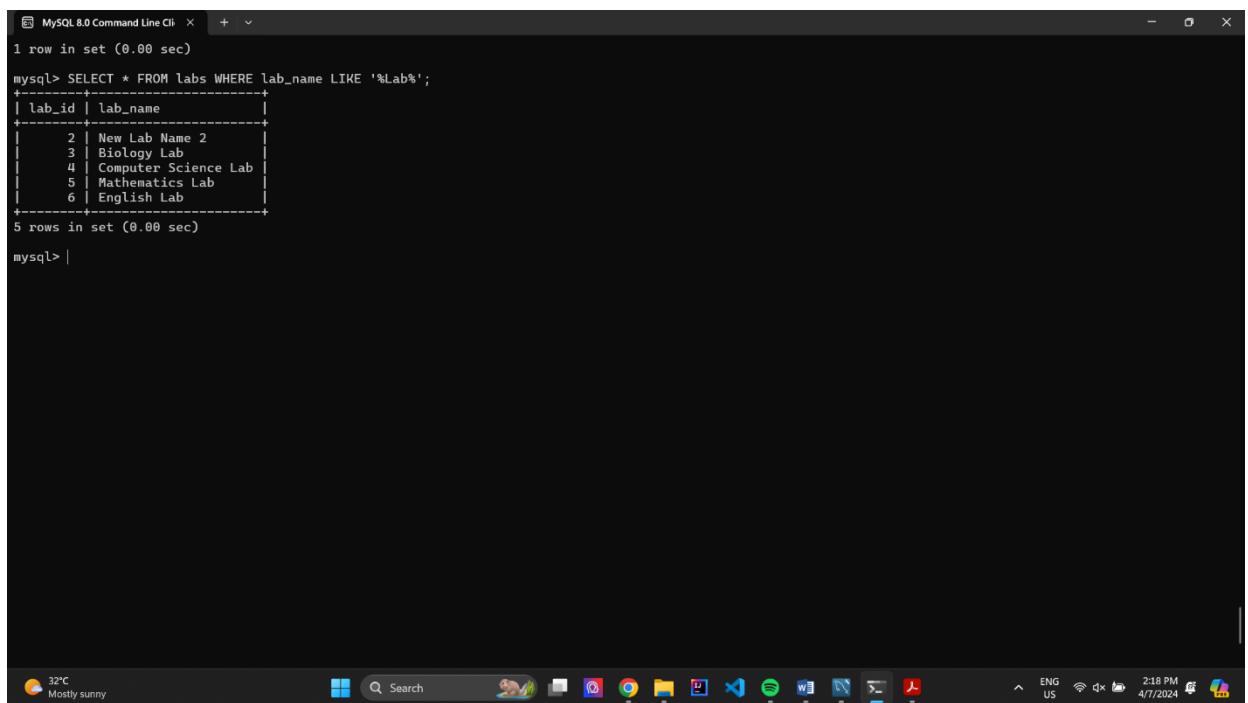
mysql> |
```

32°C Mostly sunny

Search

216 PM 4/7/2024

Figure 41 : Find the Maximum Age from Lab Maintenance Table (Aggregation Functions)



The screenshot shows a Windows desktop environment with a MySQL 8.0 Command Line Client window open. The window title is "MySQL 8.0 Command Line Cli". The command entered is "SELECT * FROM labs WHERE lab_name LIKE '%Lab%'". The output shows a table with columns "lab_id" and "lab_name". The data includes rows for New Lab Name 2, Biology Lab, Computer Science Lab, Mathematics Lab, and English Lab. The command prompt "mysql>" is visible at the bottom.

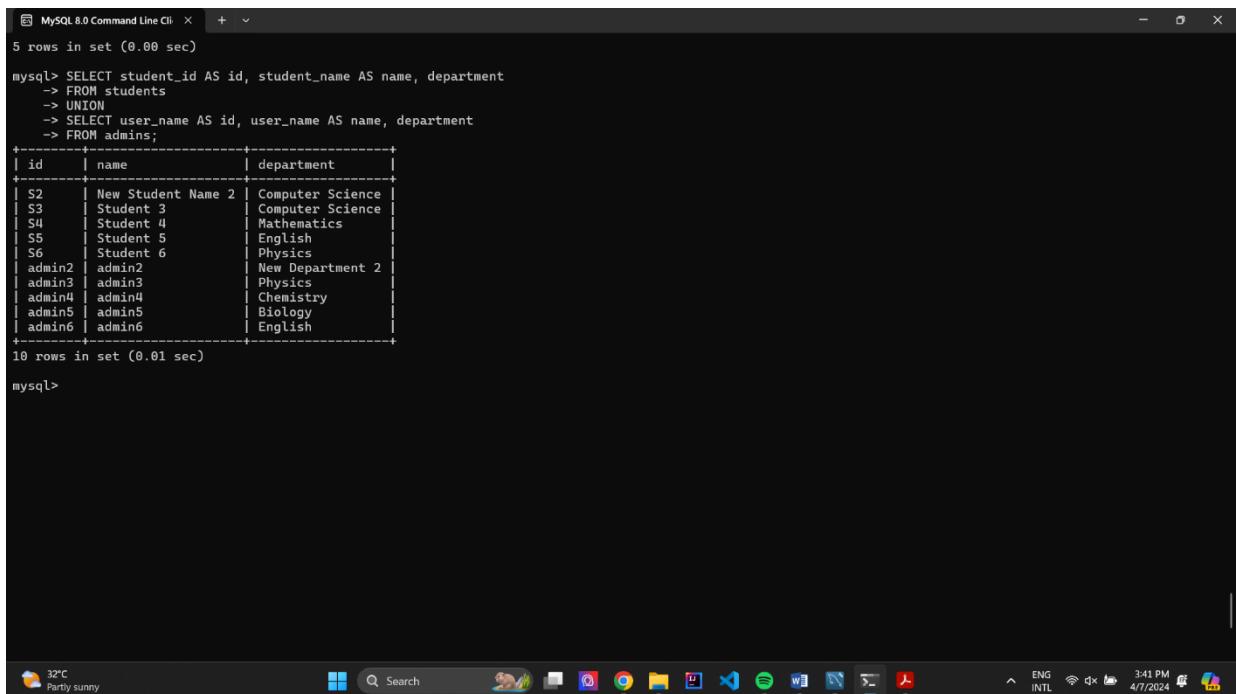
lab_id	lab_name
2	New Lab Name 2
3	Biology Lab
4	Computer Science Lab
5	Mathematics Lab
6	English Lab

5 rows in set (0.00 sec)

mysql> |

Figure 42 : Use of the keyword 'Like'

Complex Queries



MySQL 8.0 Command Line Cli + - x

```
5 rows in set (0.00 sec)

mysql> SELECT student_id AS id, student_name AS name, department
-> FROM students
-> UNION
-> SELECT user_name AS id, user_name AS name, department
-> FROM admins;
+-----+-----+-----+
| id   | name  | department |
+-----+-----+-----+
| S2   | New Student Name 2 | Computer Science |
| S3   | Student 3          | Computer Science |
| S4   | Student 4          | Mathematics    |
| S5   | Student 5          | English        |
| S6   | Student 6          | Physics         |
| admin2 | admin2 | New Department 2 |
| admin3 | admin3 | Physics        |
| admin4 | admin4 | Chemistry      |
| admin5 | admin5 | Biology        |
| admin6 | admin6 | English        |
+-----+-----+-----+
10 rows in set (0.01 sec)

mysql>
```

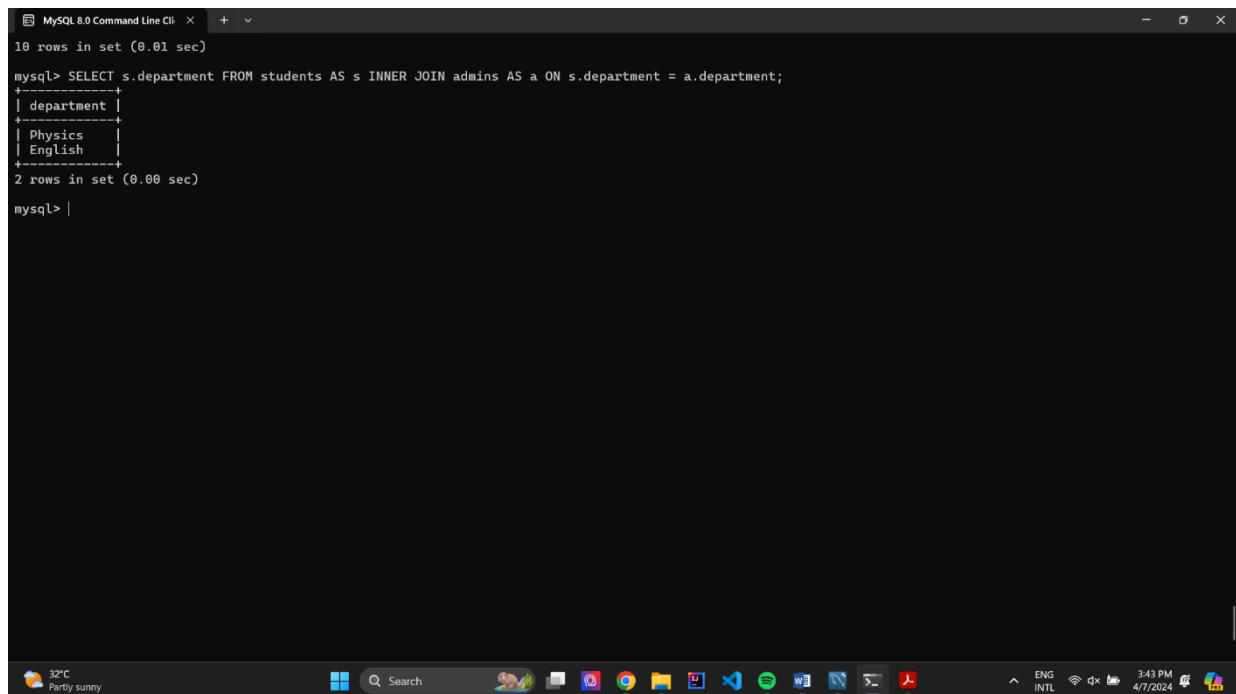
32°C Partly sunny

Search

Windows Taskbar

3:41 PM 4/7/2024

Figure 43 : Union Operation



MySQL 8.0 Command Line Cli + - x

```
10 rows in set (0.01 sec)

mysql> SELECT s.department FROM students AS s INNER JOIN admins AS a ON s.department = a.department;
+-----+
| department |
+-----+
| Physics    |
| English    |
+-----+
2 rows in set (0.00 sec)

mysql> |
```

32°C Partly sunny

Search

Windows Taskbar

3:43 PM 4/7/2024

Figure 44 : Intersection Operation

The screenshot shows a Windows desktop environment with a MySQL 8.0 Command Line Client window open. The window title is "MySQL 8.0 Command Line Cli". The command entered is:

```
mysql> SELECT l.* FROM labs AS l
-> WHERE NOT EXISTS (SELECT 2 FROM lab_equipment AS le WHERE l.lab_id = le.lab_id);
Empty set (0.00 sec)

mysql> SELECT l.* FROM labs AS l
-> WHERE NOT EXISTS (SELECT 1 FROM Labs_Maintenances AS le WHERE l.lab_id = le.lab_id);
Empty set (0.00 sec)

mysql> |
```

The system tray at the bottom left shows the weather as "32°C Partly sunny". The taskbar at the bottom right includes icons for search, file explorer, browser, and other applications. The system tray on the right shows battery status, network, and date/time.

Figure 45 : Set Difference Operation

The screenshot shows a Windows desktop environment with a MySQL 8.0 Command Line Client window open. The window title is "MySQL 8.0 Command Line Cli". The command entered is:

```
mysql> SELECT l.* FROM labs AS l
-> WHERE NOT EXISTS (SELECT 1 FROM Labs_Maintenances AS le WHERE l.lab_id = le.lab_id);
Empty set (0.00 sec)

mysql> SELECT mc.coordinator_id, mc.name FROM module_coordinator AS mc
-> WHERE NOT EXISTS ( SELECT m.module_code FROM modules AS m
-> WHERE NOT EXISTS (SELECT mc2.module_code FROM module_coordinator AS mc2
-> WHERE mc2.coordinator_id = mc.coordinator_id AND mc2.module_code = m.module_code
-> ) );
Empty set (0.00 sec)

mysql>
```

The system tray at the bottom left shows the weather as "32°C Partly sunny". The taskbar at the bottom right includes icons for search, file explorer, browser, and other applications. The system tray on the right shows battery status, network, and date/time.

Figure 46 : Division Operation

MySQL 8.0 Command Line Cli

```
mysql> INNER JOIN lab_equipment AS le ON l.lab_id = le.lab_id;
ERROR 1146 (42S02): Table 'laboratory_management_system.lab_equipment' doesn't exist
mysql> CREATE VIEW labs_and_equipment_view AS
-> SELECT l.lab_id, l.lab_name, le.equipment_id, le.ip_address
-> FROM labs AS l
-> INNER JOIN lab_equipment AS le ON l.lab_id = le.lab_id;
Query OK, 0 rows affected (0.02 sec)

mysql> SELECT * FROM labs_and_equipment_view;
+-----+-----+-----+
| lab_id | lab_name | equipment_id | ip_address |
+-----+-----+-----+
| 2 | New Lab Name 2 | 2 | 192.168.1.9 |
| 3 | Biology Lab | 3 | 192.168.1.3 |
| 4 | Computer Science Lab | 4 | 192.168.1.4 |
| 5 | Mathematics Lab | 5 | 192.168.1.5 |
| 6 | English Lab | 6 | 192.168.1.6 |
+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Figure 47 : Inner Join between the Labs and Lab Equipment tables with the user view

MySQL 8.0 Command Line Cli

```
mysql> CREATE VIEW labs_equipment_view AS
-> SELECT * FROM labs
-> NATURAL JOIN lab_equipment;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM labs_equipment_view;
+-----+-----+-----+
| lab_id | lab_name | Equipment_ID | IP_Address |
+-----+-----+-----+
| 2 | New Lab Name 2 | 2 | 192.168.1.9 |
| 3 | Biology Lab | 3 | 192.168.1.3 |
| 4 | Computer Science Lab | 4 | 192.168.1.4 |
| 5 | Mathematics Lab | 5 | 192.168.1.5 |
| 6 | English Lab | 6 | 192.168.1.6 |
+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Figure 48 : Natural Join between the Labs and Lab Equipment tables with the user view

MySQL 8.0 Command Line Cli

```
5 rows in set (0.00 sec)

mysql> CREATE VIEW student_lab_card_view AS
-> SELECT s.student_id, s.student_name, lc.lab_card_id, lc.batch_no
-> FROM students AS s
-> LEFT OUTER JOIN lab_card AS lc ON s.student_id = lc.student_id;
Query OK, 0 rows affected (0.02 sec)

mysql> SELECT * FROM student_lab_card_view;
+-----+-----+-----+
| student_id | student_name | lab_card_id | batch_no |
+-----+-----+-----+
| S2          | New Student Name 2 | 2           | 2023      |
| S3          | Student 3          | 3           | 2022      |
| S4          | Student 4          | 4           | 2021      |
| S5          | Student 5          | 5           | 2022      |
| S6          | Student 6          | 6           | 2021      |
+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Figure 49 : Left Outer Join between Student and Lab Card tables with user view

MySQL 8.0 Command Line Cli

```
RIGHT OUTER ' at line 2
mysql>
mysql> CREATE VIEW student_lab_card_right_view AS
-> SELECT s.student_id, lc.lab_card_id, lc.batch_no
-> FROM students AS s
-> RIGHT OUTER JOIN lab_card AS lc ON s.student_id = lc.student_id;
Query OK, 0 rows affected (0.02 sec)

mysql> SELECT * FROM student_lab_card_right_view;
+-----+-----+-----+
| student_id | lab_card_id | batch_no |
+-----+-----+-----+
| S2          | 2           | 2023      |
| S3          | 3           | 2022      |
| S4          | 4           | 2021      |
| S5          | 5           | 2022      |
| S6          | 6           | 2021      |
+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Figure 50 : Right Outer Join between Student and Lab Card tables with user view

```

MySQL 8.0 Command Line Cli × + ▾

mysql> CREATE VIEW student_lab_card_full_view AS
-> SELECT s.student_id, s.student_name, lc.lab_card_id, lc.batch_no
-> FROM students AS s
-> LEFT JOIN lab_card AS lc ON s.student_id = lc.student_id
-> UNION
-> SELECT s.student_id, s.student_name, lc.lab_card_id, lc.batch_no
-> FROM students AS s
-> RIGHT JOIN lab_card AS lc ON s.student_id = lc.student_id;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM student_lab_card_full_view;
+-----+-----+-----+
| student_id | student_name | lab_card_id | batch_no |
+-----+-----+-----+
| S2          | New Student Name 2 | 2           | 2023      |
| S3          | Student 3          | 3           | 2022      |
| S4          | Student 4          | 4           | 2021      |
| S5          | Student 5          | 5           | 2022      |
| S6          | Student 6          | 6           | 2021      |
+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> |

```

The screenshot shows the MySQL 8.0 Command Line Client interface. The command line displays the creation of a user-defined view named 'student_lab_card_full_view'. This view performs a full outer join between the 'students' table and the 'lab_card' table. The resulting output is a table with columns: student_id, student_name, lab_card_id, and batch_no. The data includes five rows corresponding to student IDs S2 through S6, with their respective names and lab card details.

Figure 51 : Full Outer Join between Student and Lab Card tables with user view

```

MySQL 8.0 Command Line Cli × + ▾

mysql> CREATE VIEW coordinator_assessments_view AS
-> SELECT mc.coordinator_id, mc.module_code, mc.name, mc.email, mc.department, a.assessment_id, a.student_id, a.name AS assessment_name, a.mark
-> FROM module_coordinator AS mc
-> LEFT JOIN assessments AS a ON mc.coordinator_id = a.coordinator_id
-> UNION
-> SELECT mc.coordinator_id, mc.module_code, mc.name, mc.email, mc.department, a.assessment_id, a.student_id, a.name AS assessment_name, a.mark
-> FROM module_coordinator AS mc
-> RIGHT JOIN assessments AS a ON mc.coordinator_id = a.coordinator_id;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM coordinator_assessments_view;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| coordinator_id | module_code | name       | email     | department | assessment_id | student_id | assessment_name | mark |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| C2            | CS102       | Dr. Michael Brown | newemail1@example.com | Computer Science | 2 | S2          | New Assessment Name 2 | 95   |
| C3            | MA101       | Dr. Robert Johnson | newemail2@example.com | Mathematics    | 3 | S3          | Assessment 3          | 95   |
| C4            | EN101       | Dr. Sarah Lee        | sarah.lee@gmail.com | English         | 4 | S4          | Assessment 4          | 80   |
| C5            | PH101       | Mr. David Young      | david.wang@hotmail.com | Physics         | 5 | S5          | Assessment 5          | 75   |
| C6            | CH101       | Ms. Lisa Taylor      | lisa.taylor@yahoo.com | Chemistry       | 6 | S6          | Assessment 6          | 70   |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> |

```

The screenshot shows the MySQL 8.0 Command Line Client interface. The command line displays the creation of a user-defined view named 'coordinator_assessments_view'. This view performs a full outer join between the 'module_coordinator' table and the 'assessments' table. The resulting output is a table with columns: coordinator_id, module_code, name, email, department, assessment_id, student_id, assessment_name, and mark. The data includes six rows corresponding to coordinator IDs C2 through C6, with their respective module details and assessment marks.

Figure 52 : Full Outer Join between Location and Labs tables with user view

The screenshot shows a Windows desktop environment with the MySQL 8.0 Command Line Client window open. The command line displays the creation of a view named 'students_exists_view' and its execution. The view selects all columns from the 'students' table where there exists a record in the 'lab_card' table with student_id = s.student_id and batch_no = 2021.

```
mysql> CREATE VIEW students_exists_view AS
-> SELECT * FROM students AS s
-> WHERE EXISTS (SELECT 1 FROM lab_card AS lc
-> WHERE lc.student_id = s.student_id AND lc.batch_no = 2021);
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM students_exists_view;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Student_ID | Student_Name | Semester | Password | Username | Department | Street_No | Street_Name | Town | Leads_ID |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| S4 | Student 4 | 2 | password4 | username4 | Mathematics | 321 | Street 4 | Town 4 | NULL |
| S6 | Student 6 | 2 | password6 | username6 | Physics | 987 | Street 6 | Town 6 | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

Figure 53 : Outer Union Operation with user view

The screenshot shows a Windows desktop environment with the MySQL 8.0 Command Line Client window open. The command line displays a query to select student_id and name from the assessments table where mark > 75. The results show three students: S2, S3, and S4, each associated with a specific assessment name.

```
mysql> SELECT student_id, name FROM assessments WHERE mark > 75;
+-----+-----+
| student_id | name          |
+-----+-----+
| S2         | New Assessment 2 |
| S3         | Assessment 3    |
| S4         | Assessment 4    |
+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

Figure 54 : List of Students Who Scored Above 75

The screenshot shows the MySQL 8.0 Command Line Client window. The command entered is:

```
mysql> SELECT s.student_name, a.mark FROM students AS s JOIN assessments AS a ON s.student_id = a.student_id;
```

The resulting table is:

student_name	mark
New Student Name 2	95
Student 3	95
Student 4	80
Student 5	75
Student 6	70

At the bottom of the client window, there is a status bar showing system information like temperature (31°C), weather (Partly sunny), and system time (4/7/2024 4:46 PM).

Figure 55 : Student's Assessment Marks

The screenshot shows the MySQL 8.0 Command Line Client window. The command entered is:

```
mysql> SELECT s.student_id, s.student_name  
    -> FROM students AS s JOIN assessments AS a ON s.student_id = a.student_id  
    -> WHERE a.mark > (SELECT AVG(mark) FROM assessments);
```

The resulting table is:

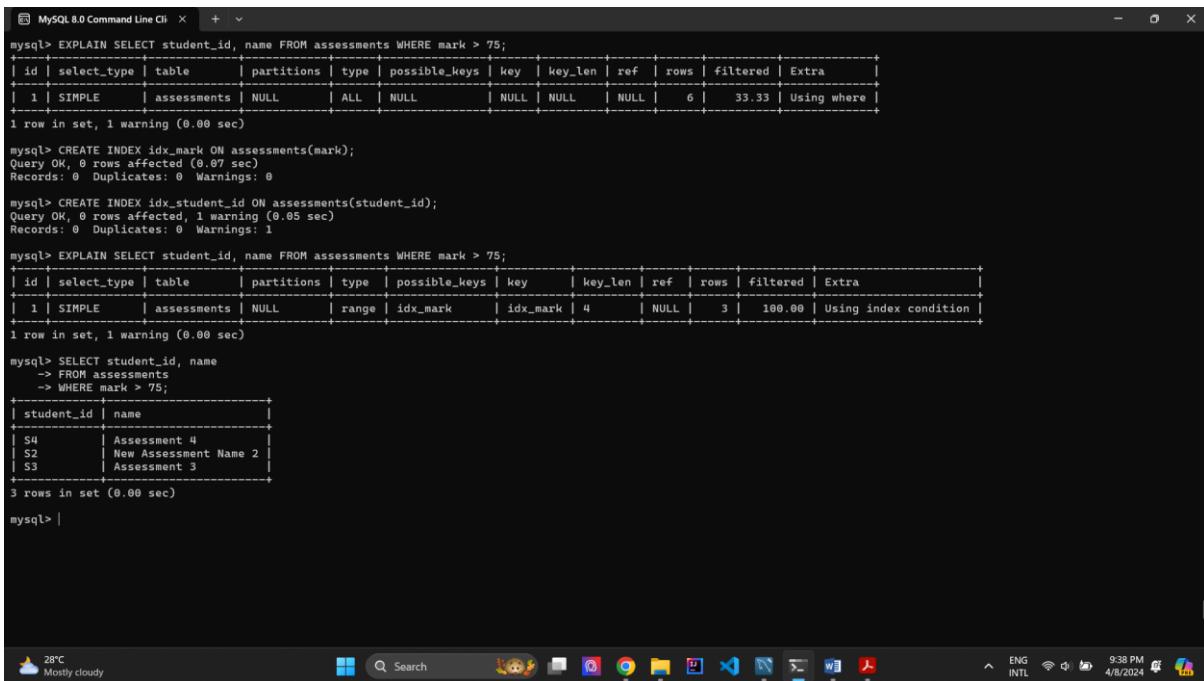
student_id	student_name
S2	New Student Name 2
S3	Student 3

At the bottom of the client window, there is a status bar showing system information like temperature (31°C), weather (Partly sunny), and system time (4/7/2024 4:50 PM).

Figure 56 : Students Who Scored Above the Average Mark

CHAPTER 5 – DATABASE TUNING

Database tuning is used to improve application performance, reduce response time and to optimize resource management. 10 queries were tuned and the difference of them with the original queries are discussed below.



The screenshot shows a MySQL 8.0 Command Line Client window. The session starts with an EXPLAIN command for a query that uses a full table scan (ALL key type). It then shows the creation of an index named idx_mark on the assessments table. Following this, another EXPLAIN command is run, which now indicates the use of the newly created index, resulting in a range scan. Finally, the actual SELECT query is executed, returning three rows of data from the assessments table where student_id is S4, S2, or S3 and name is Assessment 4, New Assessment Name 2, or Assessment 3 respectively. The MySQL client interface includes a toolbar at the top and a taskbar at the bottom with various icons and system status information.

```
mysql> EXPLAIN SELECT student_id, name FROM assessments WHERE mark > 75;
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE      | assessments | NULL    | ALL   | NULL        | NULL | NULL    | NULL |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)

mysql> CREATE INDEX idx_mark ON assessments(mark);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> CREATE INDEX idx_student_id ON assessments(student_id);
Query OK, 0 rows affected, 1 warning (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 1

mysql> EXPLAIN SELECT student_id, name FROM assessments WHERE mark > 75;
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE      | assessments | NULL    | range | idx_mark     | idx_mark | 4       | NULL |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)

mysql> SELECT student_id, name
-> FROM assessments
-> WHERE mark > 75;
+-----+-----+
| student_id | name      |
+-----+-----+
| S4          | Assessment 4 |
| S2          | New Assessment Name 2 |
| S3          | Assessment 3 |
+-----+-----+
3 rows in set (0.00 sec)

mysql> |
```

Figure 57 : Tuned Query 1 and Results Comparing

The Original Complex Query that tuned is shown below.

```
SELECT student_id, name
FROM assessments
WHERE mark > 75;
```

For the tuning process an index was created and after tuning the number of rows have been reduced 6 to 3.

```

MySQL 8.0 Command Line Cli X + v
Records: 0 Duplicates: 0 Warnings: 0
mysql> EXPLAIN SELECT * FROM student_lab_card_view;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | s | NULL | ALL | NULL | NULL | NULL | NULL |
| 1 | SIMPLE | lc | NULL | ref | Student_ID | Student_ID | 1022 | laboratory_management_system.s.Student_ID |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql> CREATE INDEX idx_student_id ON students(student_id);
Query OK, 0 rows affected (0.12 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> CREATE INDEX idx_student_id ON lab_card(student_id);
Query OK, 0 rows affected (0.08 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> EXPLAIN SELECT * FROM student_lab_card_view;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | s | NULL | ALL | NULL | NULL | NULL | NULL |
| 1 | SIMPLE | lc | NULL | ref | idx_student_id | idx_student_id | 1022 | laboratory_management_system.s.Student_ID |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql> SELECT s.student_id, s.student_name, lc.lab_card_id, lc.batch_no
-> FROM students AS s
-> LEFT OUTER JOIN lab_card AS lc USING (student_id);
+-----+-----+-----+
| student_id | student_name | lab_card_id | batch_no |
+-----+-----+-----+
| S2 | New Student Name 2 | 2 | 2023 |
| S3 | Student 3 | 3 | 2022 |
| S4 | Student 4 | 4 | 2021 |
| S5 | Student 5 | 5 | 2022 |
| S6 | Student 6 | 6 | 2021 |
+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>

```

Figure 58 : Tuned Query 2 and Results Comparing

The Original Complex Query that tuned is shown below.

```

SELECT s.student_id, s.student_name, lc.lab_card_id, lc.batch_no
FROM students AS s
LEFT OUTER JOIN lab_card AS lc USING (student_id);

```

After creating the index in Student_ID column query has utilized the indexes and the performance of the query has been improved.

```

MySQL 8.0 Command Line Cli X + v
Records: 0 Duplicates: 0 Warnings: 0
mysql> EXPLAIN SELECT * FROM student_lab_card_full_view;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | <derived2> | NULL | ALL | NULL | NULL | NULL | NULL |
| 2 | DERIVED | s | NULL | ALL | NULL | NULL | NULL | NULL |
| 2 | DERIVED | lc | NULL | ref | idx_student_id | idx_student_id | 1022 | laboratory_management_system.s.Student_ID |
| 3 | UNION | lc | NULL | ALL | NULL | NULL | NULL | NULL |
| 3 | UNION | s | NULL | eq_ref | PRIMARY | PRIMARY | 1022 | laboratory_management_system.lc.Student_ID |
| 4 | UNION RESULT | <union2,3> | NULL | ALL | NULL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set, 1 warning (0.01 sec)

mysql> CREATE INDEX idx_student_id_students ON students(student_id);
Query OK, 0 rows affected (0.10 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> CREATE INDEX idx_student_id_lab_card ON lab_card(student_id);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'INDEX idx_stu
nt_id_lab_card ON lab_card(student_id)' at line 1
mysql> create INDEX idx_student_id_lab_card ON lab_card(student_id);
Query OK, 0 rows affected, 1 warning (0.11 sec)
Records: 0 Duplicates: 0 Warnings: 1

mysql> EXPLAIN SELECT * FROM student_lab_card_full_view;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | <derived2> | NULL | ALL | NULL | NULL | NULL | NULL |
| 2 | 100.00 | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.01 sec)

mysql>

```

Figure 59 : Results for the Original query

```

| 3 | UNION      | lc      | NULL    | ALL    | NULL        | NULL      | NULL    | NULL | NULL | 6 | 100.00 | NULL
| 3 | UNION      | s       | NULL    | eq_ref | PRIMARY    | PRIMARY   | 1022   | laboratory_managment_system.lc.Student_ID | 1 | 100.00 | NULL
| 4 | UNION RESULT | <union2,3> | NULL    | ALL    | NULL        | NULL      | NULL    | NULL | NULL | NULL | NULL | Using temporary
+-----+
6 rows in set, 1 warning (0.01 sec)

mysql> CREATE INDEX idx_student_id_students ON students(student_id);
Query OK, 0 rows affected (0.10 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> INDEX idx_student_id_lab_card ON lab_card(student_id);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'INDEX idx_stu
nt_id_lab_card ON lab_card(student_id)' at line 1
mysql> create INDEX idx_student_id_lab_card ON lab_card(student_id);
Query OK, 0 rows affected, 1 warning (0.11 sec)
Records: 0 Duplicates: 0 Warnings: 1

mysql> EXPLAIN SELECT * FROM student_lab_card_full_view;
+-----+
| id | select_type | table      | partitions | type    | possible_keys          | key           | key_len | ref
+-----+
| 1  | PRIMARY     | <derived2> | NULL      | ALL    | NULL        | NULL      | NULL    | NULL
| 2  | DERIVED     | s          | NULL      | ALL    | NULL        | NULL      | NULL    | NULL
| 5  | UNION        | lc         | NULL      | ref    | idx_student_id, idx_student_id_lab_card | idx_student_id | 1022 | laboratory_managment_system.s.Student_ID |
| 2  | DERIVED     | lc         | NULL      | ALL    | NULL        | NULL      | NULL    | NULL
| 1  | UNION        | lc         | NULL      | ALL    | NULL        | NULL      | NULL    | NULL
| 3  | UNION        | s          | NULL      | eq_ref | PRIMARY, idx_student_id_students | PRIMARY | 1022 | laboratory_managment_system.lc.Student_ID |
| 6  | UNION        | s          | NULL      | ALL    | NULL        | NULL      | NULL    | NULL
| 1  | UNION        | s          | NULL      | eq_ref | PRIMARY, idx_student_id_students | PRIMARY | 1022 | laboratory_managment_system.lc.Student_ID |
| 4  | UNION RESULT | <union2,3> | NULL      | ALL    | NULL        | NULL      | NULL    | NULL
+-----+
ULL | NULL | Using temporary
+-----+
6 rows in set, 1 warning (0.00 sec)

mysql> |

```

Figure 60 : Results of the Tuned query

```

Records: 0 Duplicates: 0 Warnings: 0

mysql> INDEX idx_student_id_lab_card ON lab_card(student_id);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'INDEX idx_stu
nt_id_lab_card ON lab_card(student_id)' at line 1
mysql> create INDEX idx_student_id_lab_card ON lab_card(student_id);
Query OK, 0 rows affected, 1 warning (0.11 sec)
Records: 0 Duplicates: 0 Warnings: 1

mysql> EXPLAIN SELECT * FROM student_lab_card_full_view;
+-----+
| id | select_type | table      | partitions | type    | possible_keys          | key           | key_len | ref
+-----+
| 1  | PRIMARY     | <derived2> | NULL      | ALL    | NULL        | NULL      | NULL    | NULL
| 2  | DERIVED     | s          | NULL      | ALL    | NULL        | NULL      | NULL    | NULL
| 5  | UNION        | lc         | NULL      | ref    | idx_student_id, idx_student_id_lab_card | idx_student_id | 1022 | laboratory_managment_system.s.Student_ID |
| 2  | DERIVED     | lc         | NULL      | ALL    | NULL        | NULL      | NULL    | NULL
| 1  | UNION        | lc         | NULL      | ALL    | NULL        | NULL      | NULL    | NULL
| 6  | UNION        | s          | NULL      | eq_ref | PRIMARY, idx_student_id_students | PRIMARY | 1022 | laboratory_managment_system.lc.Student_ID |
| 3  | UNION        | s          | NULL      | eq_ref | PRIMARY, idx_student_id_students | PRIMARY | 1022 | laboratory_managment_system.lc.Student_ID |
| 1  | UNION        | s          | NULL      | ALL    | NULL        | NULL      | NULL    | NULL
| 4  | UNION RESULT | <union2,3> | NULL      | ALL    | NULL        | NULL      | NULL    | NULL
+-----+
ULL | NULL | Using temporary
+-----+
6 rows in set, 1 warning (0.00 sec)

mysql> SELECT * FROM student_lab_card_full_view;
+-----+
| student_id | student_name | lab_card_id | batch_no |
+-----+
| S2 | New Student Name 2 | 2 | 2023 |
| S3 | Student 3 | 3 | 2022 |
| S4 | Student 4 | 4 | 2021 |
| S5 | Student 5 | 5 | 2022 |
| S6 | Student 6 | 6 | 2021 |
+-----+
5 rows in set (0.00 sec)

mysql> |

```

Figure 61 : Tuned Query 3

The Original Complex Query that tuned is shown below.

```

CREATE VIEW student_lab_card_full_view AS
SELECT s.student_id, s.student_name, lc.lab_card_id, lc.batch_no
FROM students AS s
LEFT JOIN lab_card AS lc ON s.student_id = lc.student_id
UNION

```

```

SELECT s.student_id, s.student_name, lc.lab_card_id, lc.batch_no
FROM students AS s
RIGHT JOIN lab_card AS lc ON s.student_id = lc.student_id;
SELECT * FROM student_lab_card_full_view;

```

After creating the index in Student_ID column query has utilized the indexes and the performance of the query has been improved.

```

mysql> EXPLAIN SELECT * FROM coordinator_assessments_view;
+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys |
+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | <derived2> | NULL | ALL | NULL |
| 2 | DERIVED | mc | NULL | ALL | NULL |
| 2 | DERIVED | a | NULL | ref | idx_assessments_coordinator_id |
| 3 | UNION | a | NULL | ALL | NULL |
| 3 | UNION | mc | NULL | eq_ref | PRIMARY |
| 4 | UNION RESULT | <union2_3> | NULL | ALL | NULL |
+-----+-----+-----+-----+-----+-----+
6 rows in set, 1 warning (0.00 sec)

mysql> CREATE INDEX idx_coordinator_id_module_coordinator ON module_coordinator(coordinator_id);
Query OK, 0 rows affected (0.08 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> drop INDEX idx_assessments_coordinator_id ON assessments(coordinator_id);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '(coordinator_id)' at line 1
mysql> drop INDEX idx_assessments_coordinator_id ON assessments;
ERROR 1553 (HY000): Cannot drop index 'idx_assessments_coordinator_id': needed in a foreign key constraint
mysql> EXPLAIN SELECT * FROM coordinator_assessments_view;
+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys |
+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | <derived2> | NULL | ALL | NULL |
| 2 | DERIVED | mc | NULL | ALL | NULL |
| 2 | DERIVED | a | NULL | ref | idx_assessments_coordinator_id |
+-----+-----+-----+-----+-----+-----+
6 rows in set, 1 warning (0.00 sec)

27°C Partly cloudy   Search   ENG INTL  11:29 PM  4/8/2024

```

Figure 63 : Results of Original query

```

mysql> EXPLAIN SELECT * FROM coordinator_assessments_view;
+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys |
+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | <derived2> | NULL | ALL | NULL |
| 2 | DERIVED | mc | NULL | ALL | NULL |
| 2 | DERIVED | a | NULL | ref | idx_assessments_coordinator_id |
| 3 | UNION | a | NULL | ALL | NULL |
| 3 | UNION | mc | NULL | eq_ref | PRIMARY, idx_coordinator_id_module_coordinator |
| 4 | UNION RESULT | <union2_3> | NULL | ALL | NULL |
+-----+-----+-----+-----+-----+-----+
6 rows in set, 1 warning (0.00 sec)

mysql> CREATE INDEX idx_coordinator_id_module_coordinator ON module_coordinator(coordinator_id);
Query OK, 0 rows affected (0.08 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> drop INDEX idx_assessments_coordinator_id ON assessments(coordinator_id);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '(coordinator_id)' at line 1
mysql> drop INDEX idx_assessments_coordinator_id ON assessments;
ERROR 1553 (HY000): Cannot drop index 'idx_assessments_coordinator_id': needed in a foreign key constraint
mysql> EXPLAIN SELECT * FROM coordinator_assessments_view;
+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys |
+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | <derived2> | NULL | ALL | NULL |
| 2 | DERIVED | mc | NULL | ALL | NULL |
| 2 | DERIVED | a | NULL | ref | idx_assessments_coordinator_id |
| 3 | UNION | a | NULL | ALL | NULL |
| 3 | UNION | mc | NULL | eq_ref | PRIMARY, idx_coordinator_id_module_coordinator |
| 4 | UNION RESULT | <union2_3> | NULL | ALL | NULL |
+-----+-----+-----+-----+-----+-----+
6 rows in set, 1 warning (0.00 sec)

mysql> SELECT * FROM coordinator_assessments_view;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| coordinator_id | module_code | name | email | department | assessment_id | student_id | assessment_name | mark |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| C2 | CS102 | Dr. Michael Brown | newemail@example.com | Computer Science | 2 | S2 | New Assessment Name 2 | 95 |
| C3 | MA101 | Dr. Robert Johnson | newemail2@example.com | Mathematics | 3 | S3 | Assessment 3 | 95 |
| C4 | EN101 | Dr. Sarah Lee | sarah.lee@gmail.com | English | 4 | S4 | Assessment 4 | 88 |
| C5 | PH101 | Mr. David Young | david.wang@hotmail.com | Physics | 5 | S5 | Assessment 5 | 75 |
| C6 | CH101 | Ms. Lisa Taylor | lisa.taylor@yahoo.com | Chemistry | 6 | S6 | Assessment 6 | 78 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
27°C Partly cloudy   Search   ENG INTL  11:30 PM  4/8/2024

```

Figure 62 : Tuned query 4 and Results of Tuned query

The Original Complex Query that tuned is shown below.

```
CREATE VIEW coordinator_assessments_view AS
SELECT mc.coordinator_id, mc.module_code, mc.name, mc.email, mc.department, a.assessment_id,
a.student_id, a.name AS assessment_name, a.mark
FROM module_coordinator AS mc
LEFT JOIN assessments AS a ON mc.coordinator_id = a.coordinator_id
UNION
SELECT mc.coordinator_id, mc.module_code, mc.name, mc.email, mc.department, a.assessment_id,
a.student_id, a.name AS assessment_name, a.mark
FROM module_coordinator AS mc
RIGHT JOIN assessments AS a ON mc.coordinator_id = a.coordinator_id;
SELECT * FROM coordinator_assessments_view;
```

After creating the index in Coordinator ID column query has utilized the indexes and the performance of the query has been improved.

The screenshot shows a MySQL command-line interface window. It displays the creation of a view named 'students_exists_view' and its tuned version 'students_exists_view_tuned'. The 'EXPLAIN' command is used to show the execution plans for both queries, highlighting the use of indexes. The final 'SELECT *' command shows the results of the tuned query, which includes columns: Student_ID, Student_Name, Semester, Password, Username, Department, Street_No, Street_Name, Town, and Leadr_ID. The results show two rows: one for Student_ID 54 (Student 4) and one for Student_ID 56 (Student 6).

```
mysql> explain SELECT * FROM students_exists_view;
+----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys |
+----+-----+-----+-----+-----+-----+
| 1 | SIMPLE     | s      | NULL      | ALL  | PRIMARY, idx_student_id_students |
| 1 | SIMPLE     | lc     | NULL      | ALL  | idx_student_id, idx_student_id_lab_card |
+----+-----+-----+-----+-----+-----+
2 rows in set, 2 warnings (0.00 sec)

mysql> CREATE VIEW students_exists_view_tuned AS
-> SELECT s.* 
-> FROM students AS s
-> JOIN lab_card AS lc ON s.student_id = lc.student_id
-> WHERE lc.batch_no = 2021;
Query OK, 0 rows affected (0.01 sec)

mysql> Explain Select * FROM students_exists_view_tuned;
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE     | lc    | NULL      | ALL  | idx_student_id, idx_student_id_lab_card | NULL | NULL   | NULL |
| 1 | SIMPLE     | s     | NULL      | eq_ref| PRIMARY, idx_student_id_students | PRIMARY | 1022  | laboratory_management_system.lc.Student_ID |
| 0 | NULL       |      |           |       |             |       |       |       |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql> Select * FROM students_exists_view_tuned;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Student_ID | Student_Name | Semester | Password | Username | Department | Street_No | Street_Name | Town | Leadr_ID |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 54          | Student 4    | 2         | password4 | username4 | Mathematics | 321       | Street 4    | Town 4    | NULL      |
| 56          | Student 6    | 2         | password6 | username6 | Physics     | 987       | Street 6    | Town 6    | NULL      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

Figure 64 : Tuned query 5 with the Results Comparing

The Original Complex Query that tuned is shown below.

```
CREATE VIEW students_exists_view AS
SELECT * FROM students AS s
WHERE EXISTS (
    SELECT 1 FROM lab_card AS lc
    WHERE lc.student_id = s.student_id AND lc.batch_no = 2021
);
SELECT * FROM students_exists_view;
```

The Tuning was done using JOIN was used instead of EXISTS. From that the number of rows have been reduced and query performance has been improved.

```

mysql> EXPLAIN
--> SELECT student_id AS id, student_name AS name, department
--> FROM students
--> UNION
--> SELECT user_name AS id, user_name AS name, department
--> FROM admins;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | students | NULL | ALL | NULL | NULL | NULL | NULL | 5 |
| 2 | UNION | admins | NULL | ALL | NULL | NULL | NULL | NULL | 5 |
| 3 | UNION RESULT | <union1,2> | NULL | ALL | NULL | NULL | NULL | NULL | 5 |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set, 1 warning (0.00 sec)

mysql> SELECT student_id AS id, student_name AS name, department
--> FROM students
--> UNION ALL
--> SELECT user_name AS id, user_name AS name, department
--> FROM admins;
+----+-----+-----+
| id | name | department |
+----+-----+-----+
| S2 | New Student Name 2 | Computer Science |
| S3 | Student 3 | Computer Science |
| S4 | Student 4 | Mathematics |
| S5 | Student 5 | English |
| S6 | Student 6 | Physics |
| admin2 | admin2 | New Department 2 |
| admin3 | admin3 | Physics |
| admin4 | admin4 | Chemistry |
| admin5 | admin5 | Biology |
| admin6 | admin6 | English |
+----+-----+-----+
10 rows in set (0.00 sec)

mysql> EXPLAIN
--> SELECT student_id AS id, student_name AS name, department
--> FROM students
--> UNION ALL
--> SELECT user_name AS id, user_name AS name, department
--> FROM admins;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

mysql> EXPLAIN
--> SELECT student_id AS id, student_name AS name, department
--> FROM students
--> UNION ALL
--> SELECT user_name AS id, user_name AS name, department
--> FROM admins;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | students | NULL | ALL | NULL | NULL | NULL | NULL | 5 |
| 2 | UNION | admins | NULL | ALL | NULL | NULL | NULL | NULL | 5 |
| 3 | UNION RESULT | <union1,2> | NULL | ALL | NULL | NULL | NULL | NULL | 5 |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set, 1 warning (0.00 sec)

mysql> SELECT student_id AS id, student_name AS name, department
--> FROM students
--> UNION ALL
--> SELECT user_name AS id, user_name AS name, department
--> FROM admins;
+----+-----+-----+
| id | name | department |
+----+-----+-----+
| S2 | New Student Name 2 | Computer Science |
| S3 | Student 3 | Computer Science |
| S4 | Student 4 | Mathematics |
| S5 | Student 5 | English |
| S6 | Student 6 | Physics |
| admin2 | admin2 | New Department 2 |
| admin3 | admin3 | Physics |
| admin4 | admin4 | Chemistry |
| admin5 | admin5 | Biology |
| admin6 | admin6 | English |
+----+-----+-----+
10 rows in set (0.00 sec)

mysql> EXPLAIN
--> SELECT student_id AS id, student_name AS name, department
--> FROM students
--> UNION ALL
--> SELECT user_name AS id, user_name AS name, department
--> FROM admins;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | students | NULL | ALL | NULL | NULL | NULL | NULL | 5 |
| 2 | UNION | admins | NULL | ALL | NULL | NULL | NULL | NULL | 5 |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql>

```

Figure 65 : Tuned query 6 and Results Comparing

The Original Complex Query that tuned is shown below.

```

SELECT student_id AS id, student_name AS name, department
FROM students
UNION
SELECT user_name AS id, user_name AS name, department
FROM admins;

```

The Tuning was done using UNION ALL was used instead of UNION. From that the number of rows have been reduced and query performance has been improved

```

MySQL 8.0 Command Line Cli  x  +  v
2 rows in set, 1 warning (0.00 sec)

mysql> EXPLAIN SELECT s.department FROM students AS s INNER JOIN admins AS a ON s.department = a.department;
+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys      | key           | key_len | ref
+-----+-----+-----+-----+-----+
| 1 | SIMPLE     | s     | NULL       | index | idx_students_department | idx_students_department | 1022    | NULL
| 1 | SIMPLE     | a     | NULL       | ref   | idx_admins_department  | idx_admins_department | 1022    | laboratory_management_system.s.Department |
+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql> SELECT DISTINCT s.department
   -> FROM students AS s
   -> WHERE EXISTS (
   ->   SELECT 1 FROM admins AS a WHERE s.department = a.department);
+-----+
| department |
+-----+
| English   |
| Physics   |
+-----+
2 rows in set (0.00 sec)

mysql> EXPLAIN SELECT DISTINCT s.department FROM students AS s WHERE EXISTS (SELECT 1 FROM admins AS a WHERE s.department = a.department);
+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys      | key           | key_len | ref
+-----+-----+-----+-----+-----+
| 1 | SIMPLE     | s     | NULL       | index | idx_students_department | idx_students_department | 1022    | NULL
| 1 | SIMPLE     | a     | NULL       | ref   | idx_admins_department  | idx_admins_department | 1022    | laboratory_management_system.s.Department |
+-----+-----+-----+-----+-----+
2 rows in set, 2 warnings (0.00 sec)

mysql> |

```

Figure 66 : Tuned query 7 with Results Comparing

The Original Complex Query that tuned is shown below.

```

SELECT s.department
FROM students AS s
INNER JOIN admins AS a ON s.department = a.department;

```

The Tuning was done using DISTINCT key word. From that query performance has been improved

```

MySQL 8.0 Command Line Cli  x  +  v
2 rows in set, 2 warnings (0.00 sec)

mysql> EXPLAIN SELECT le.lab_id FROM lab_equipment AS le WHERE NOT EXISTS (SELECT i.issue FROM issues AS i WHERE NOT EXISTS ( SELECT le2.equipment_id FROM lab_equipment AS le2 INNER JOIN issues AS i2 ON le2.equipment_id = i2.equipment_id WHERE le2.lab_id = le.lab_id AND i2.issue = i.issue ));
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys      | key           | key_len | ref
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY     | le    | NULL       | index | NULL             | idx_lab_id   | 4        | NULL
| 2 | DEPENDENT SUBQUERY | i    | NULL       | Using where; Using index | ALL          | NULL         | NULL     | NULL
| 2 | DEPENDENT SUBQUERY | <subquery3> | NULL       | eq_ref  | <auto_distinct_key> | <auto_distinct_key> | 1028    | laboratory_management_system.le.Lab_ID, laboratory_man
agent_system.i.Issue |
| 3 | MATERIALIZED   | le2   | NULL       | ref    | PRIMARY, idx_lab_id | idx_lab_id   | 4        | laboratory_management_system.le.Lab_ID
| 3 | MATERIALIZED   | i2    | NULL       | ref    | Equipment_ID      | Equipment_ID | 4        | laboratory_management_system.le2.Equipment_ID
+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set, 3 warnings (0.01 sec)

mysql> SELECT DISTINCT le.lab_id
   -> FROM lab_equipment AS le
   -> LEFT JOIN issues AS i ON le.equipment_id = i.equipment_id WHERE i.issue IS NULL;
Empty set (0.00 sec)

mysql> EXPLAIN SELECT DISTINCT le.lab_id FROM lab_equipment AS le LEFT JOIN issues AS i ON le.equipment_id = i.equipment_id WHERE i.issue IS NULL;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys      | key           | key_len | ref
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE     | le    | NULL       | index | idx_lab_id       | idx_lab_id   | 4        | NULL
| 1 | SIMPLE     | i    | NULL       | ALL    | Equipment_ID     | NULL         | NULL     | NULL
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql> |

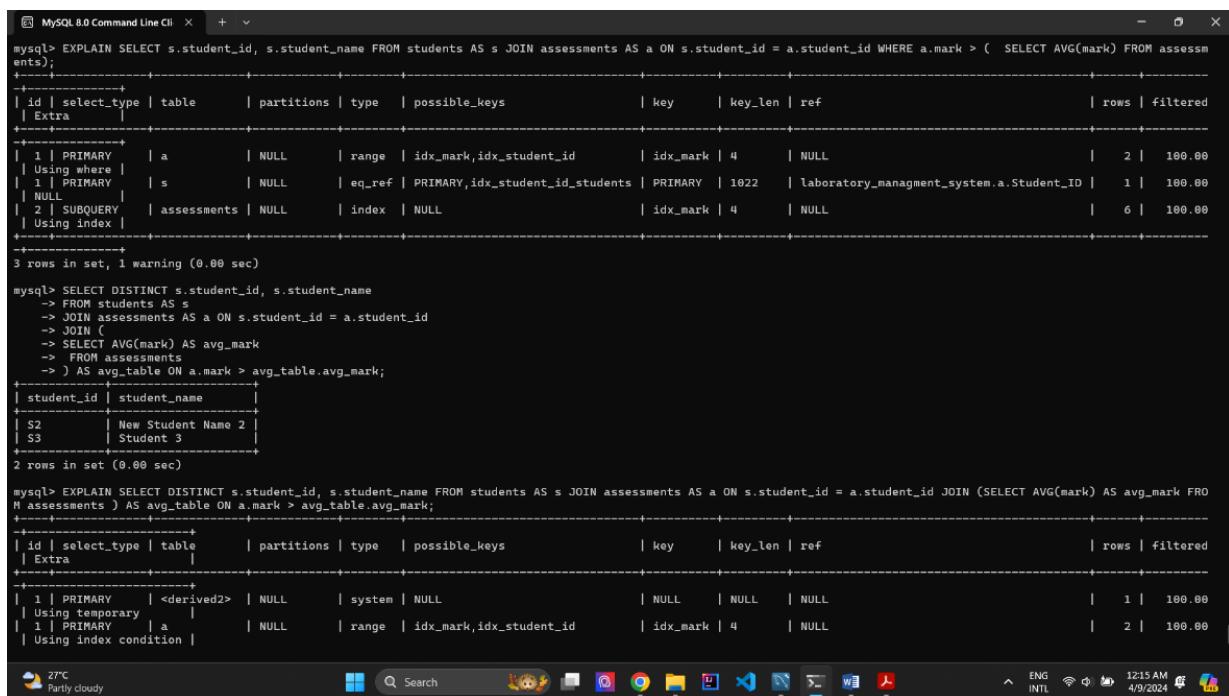
```

Figure 67 : Tuned query 8 with Results Comparing

The Original Complex Query that tuned is shown below.

```
SELECT le.lab_id
FROM lab_equipment AS le
WHERE NOT EXISTS (
    SELECT i.issue
    FROM issues AS i
    WHERE NOT EXISTS (
        SELECT le2.equipment_id
        FROM lab_equipment AS le2
        INNER JOIN issues AS i2 ON le2.equipment_id = i2.equipment_id
        WHERE le2.lab_id = le.lab_id AND i2.issue = i.issue
    )
);
```

The tuning processing was done using LEFT JOIN and removing the nested sub queries. From that the number of rows have been reduced and query performance has been improved.



The screenshot shows a MySQL command-line interface window. It contains two main sections: the first section shows the original complex query and its EXPLAIN output, while the second section shows the tuned query and its EXPLAIN output.

Original Complex Query (Top):

```
mysql> EXPLAIN SELECT s.student_id, s.student_name FROM students AS s JOIN assessments AS a ON s.student_id = a.student_id WHERE a.mark > ( SELECT AVG(mark) FROM assessments );
+----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys |
+----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | a | NULL | range | idx_mark, idx_student_id |
| 1 | PRIMARY | s | NULL | eq_ref | PRIMARY, idx_student_id |
| 2 | SUBQUERY | assessments | NULL | index | NULL |
+----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | a | NULL | range | idx_mark, idx_student_id |
| 1 | PRIMARY | s | NULL | eq_ref | PRIMARY, idx_student_id |
| 1 | PRIMARY | assessments | NULL | index | idx_mark |
+----+-----+-----+-----+-----+-----+
```

Tuned Query (Bottom):

```
mysql> EXPLAIN SELECT DISTINCT s.student_id, s.student_name
    > FROM students AS s
    > JOIN assessments AS a ON s.student_id = a.student_id
    > JOIN (
    >     SELECT AVG(mark) AS avg_mark
    >     FROM assessments
    > ) AS avg_table ON a.mark > avg_table.avg_mark;
+-----+-----+-----+-----+-----+
| student_id | student_name |
+-----+-----+
| S2 | New Student Name 2 |
| S3 | Student 3 |
+-----+-----+
```

EXPLAIN Output for Tuned Query:

```
mysql> EXPLAIN SELECT DISTINCT s.student_id, s.student_name FROM students AS s JOIN assessments AS a ON s.student_id = a.student_id JOIN (SELECT AVG(mark) AS avg_mark FROM assessments ) AS avg_table ON a.mark > avg_table.avg_mark;
+----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys |
+----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | <derived2> | NULL | system | NULL |
| 1 | PRIMARY | a | NULL | range | idx_mark, idx_student_id |
+----+-----+-----+-----+-----+-----+
```

```

MySQL 8.0 Command Line Cli  X + v

| Using where |
| 1 | PRIMARY   | s           | NULL      | eq_ref  | PRIMARY, idx_student_id_students | PRIMARY | 1022    | laboratory_management_system.a.Student_ID | 1 | 100.00
| 2 | SUBQUERY   | assessments | NULL      | index   | idx_mark | 4       | NULL
+-----+-----+-----+-----+-----+-----+-----+-----+
| Using index |
+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set, 1 warning (0.00 sec)

mysql> SELECT DISTINCT s.student_id, s.student_name
-> FROM students AS s
-> JOIN assessments AS a ON s.student_id = a.student_id
-> JOIN (
->   SELECT AVG(mark) AS avg_mark
->   FROM assessments
-> ) AS avg_table ON a.mark > avg_table.avg_mark;
+-----+-----+
| student_id | student_name |
+-----+-----+
| S2          | New Student Name 2 |
| S3          | Student 3          |
+-----+-----+
2 rows in set (0.00 sec)

mysql> EXPLAIN SELECT DISTINCT s.student_id, s.student_name FROM students AS s JOIN assessments AS a ON s.student_id = a.student_id JOIN (SELECT AVG(mark) AS avg_mark FRO
M assessments ) AS avg_table ON a.mark > avg_table.avg_mark;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table      | partitions | type    | possible_keys      | key      | key_len | ref
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY     | <derived2> | NULL      | system  | NULL             | NULL    | NULL    | NULL
| 1 | PRIMARY     | a           | NULL      | range   | idx_mark, idx_student_id | idx_mark | 4       | NULL
| 1 | PRIMARY     | s           | NULL      | eq_ref  | PRIMARY, idx_student_id_students | PRIMARY | 1022    | laboratory_management_system.a.Student_ID | 1 | 100.00
| 2 | DERIVED     | assessments | NULL      | index   | idx_mark | 4       | NULL
| 2 | DERIVED     | assessments | NULL      | index   | idx_mark | 4       | NULL
+-----+-----+-----+-----+-----+-----+-----+-----+
4 rows in set, 1 warning (0.00 sec)
mysql> |

```

Figure 68 : Tuned query 9 with Results Comparing

The Original Complex Query that tuned is shown below.

```

SELECT s.student_id, s.student_name
FROM students AS s
JOIN assessments AS a ON s.student_id = a.student_id
WHERE a.mark > (
  SELECT AVG(mark)
  FROM assessments
);

```

From this tuned query it avoids the nested sub queries in the WHERE clause.

The screenshot shows the MySQL 8.0 Command Line Client interface. The command line displays two sets of queries and their EXPLAIN outputs. The first set shows an initial query with a syntax error and its EXPLAIN output. The second set shows a tuned query using LEFT JOIN and its EXPLAIN output, which indicates better performance (1 row vs 5 rows). The system tray at the bottom shows the date and time as 4/9/2024 12:25 AM.

```

mysql> EXPLAIN SELECT l.* FROM labs AS l WHERE NOT EXISTS (SELECT 1 FROM lab_equipment AS le WHERE l.lab_id = le.lab_id);
+----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref
+----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE     | l      | NULL      | ALL   | NULL        | NULL | NULL    | NULL
| 1 | SIMPLE     | le     | NULL      | ref   | idx_lab_id  | idx_lab_id | 4    | laboratory_management_system.l.lab_id
Using index
+----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 2 warnings (0.00 sec)

mysql> SELECT l.* FROM labs AS l
-> LEFT JOIN lab_equipment AS le ON l.lab_id = le.lab_id
-> EXPLAIN SELECT l.* FROM labs AS l WHERE NOT EXISTS (SELECT 1 FROM lab_equipment AS le WHERE l.lab_id = le.lab_id);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'EXPLAIN SELECT l.* FROM labs AS l WHERE NOT EXISTS (SELECT 1 FROM lab_equipment' at line 4
mysql> SELECT l.* FROM labs AS l
-> LEFT JOIN lab_equipment AS le ON l.lab_id = le.lab_id
-> WHERE le.lab_id IS NULL;
Empty set (0.00 sec)

mysql> EXPLAIN SELECT l.* FROM labs AS l LEFT JOIN lab_equipment AS le ON l.lab_id = le.lab_id WHERE le.lab_id IS NULL;
+----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref
+----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE     | l      | NULL      | ALL   | NULL        | NULL | NULL    | NULL
| 1 | SIMPLE     | le     | NULL      | ref   | idx_lab_id  | idx_lab_id | 4    | laboratory_management_system.l.lab_id
Using index
+----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql>

```

Figure 69 : Tuned query 10 with Results Comparing

The Original Complex Query that tuned is shown below.

```

SELECT l.*
FROM labs AS l
WHERE NOT EXISTS (
  SELECT 1
  FROM lab_equipment AS le
  WHERE l.lab_id = le.lab_id
);

```

Since the tuned query uses LEFT JOIN and avoids NOT EXISTS it performs better especially with large datasets.

REFERENCES

1. "What are the 7 phases of database design?," DatabaseTown, <https://databasetown.com/what-are-the-7-phases-of-database-design/> (accessed Mar. 31, 2024).
2. "Performance Tuning SQL Queries | Advanced SQL - Mode Analytics," Mode Resources, May 23, 2016. <https://mode.com/sql-tutorial/sql-performance-tuning>