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

Table of Figures

INTRODUCTION

Database systems serves 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

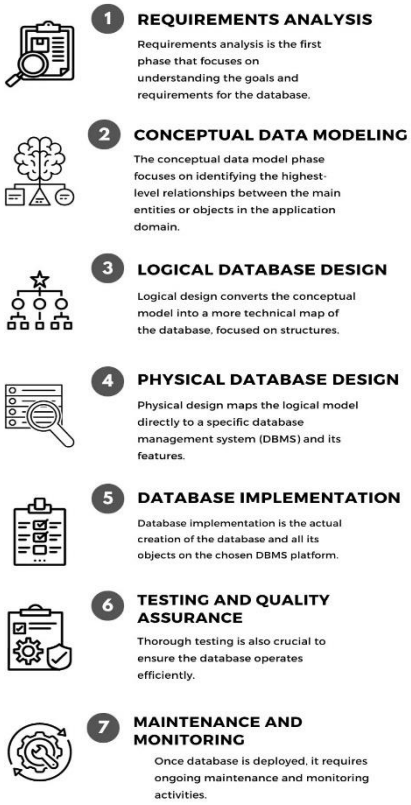


Figure 1 : Main Phases of Databases Designing [1]

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.

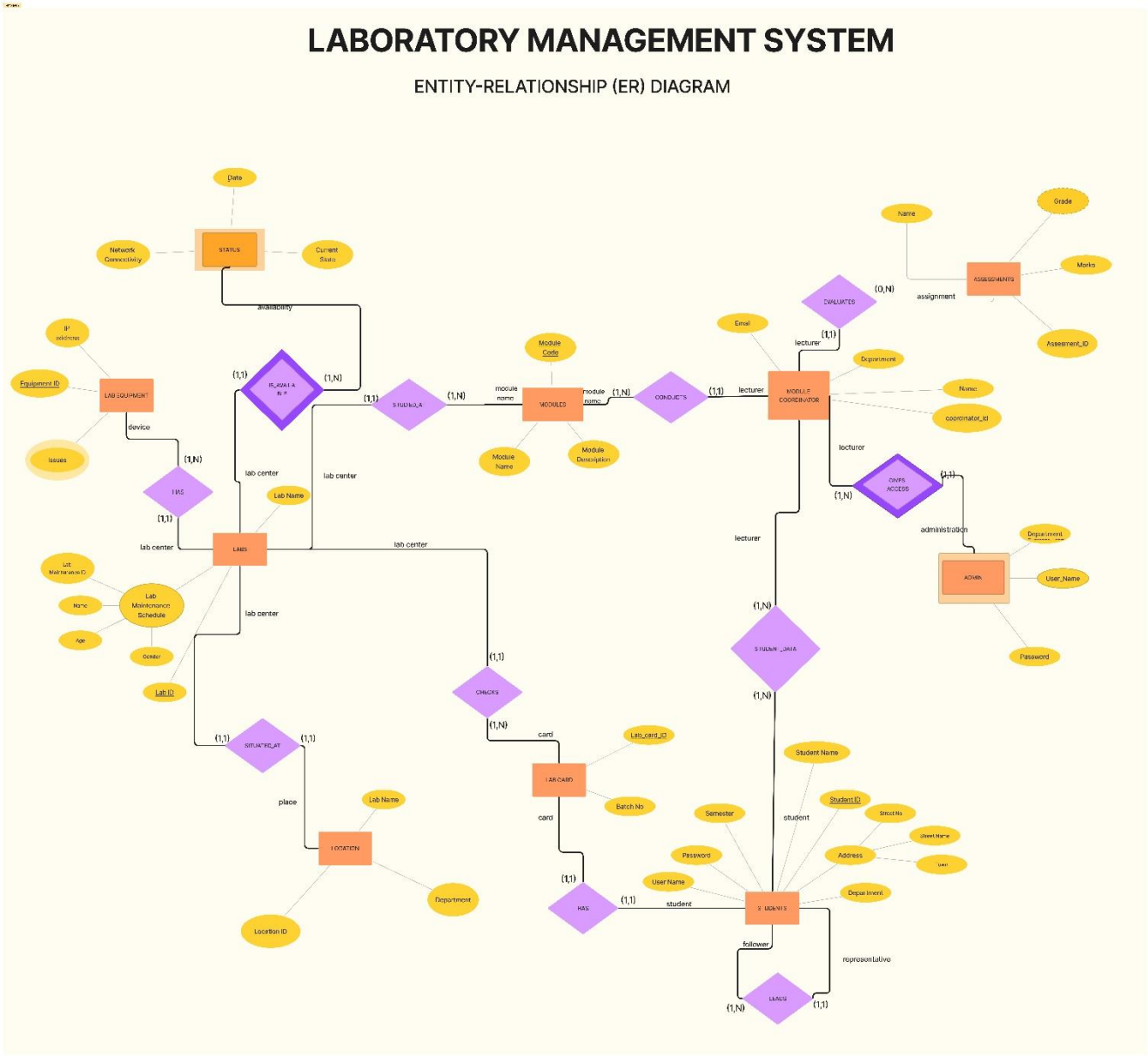


Figure 2 : ER Diagram for the Laboratory Management System

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

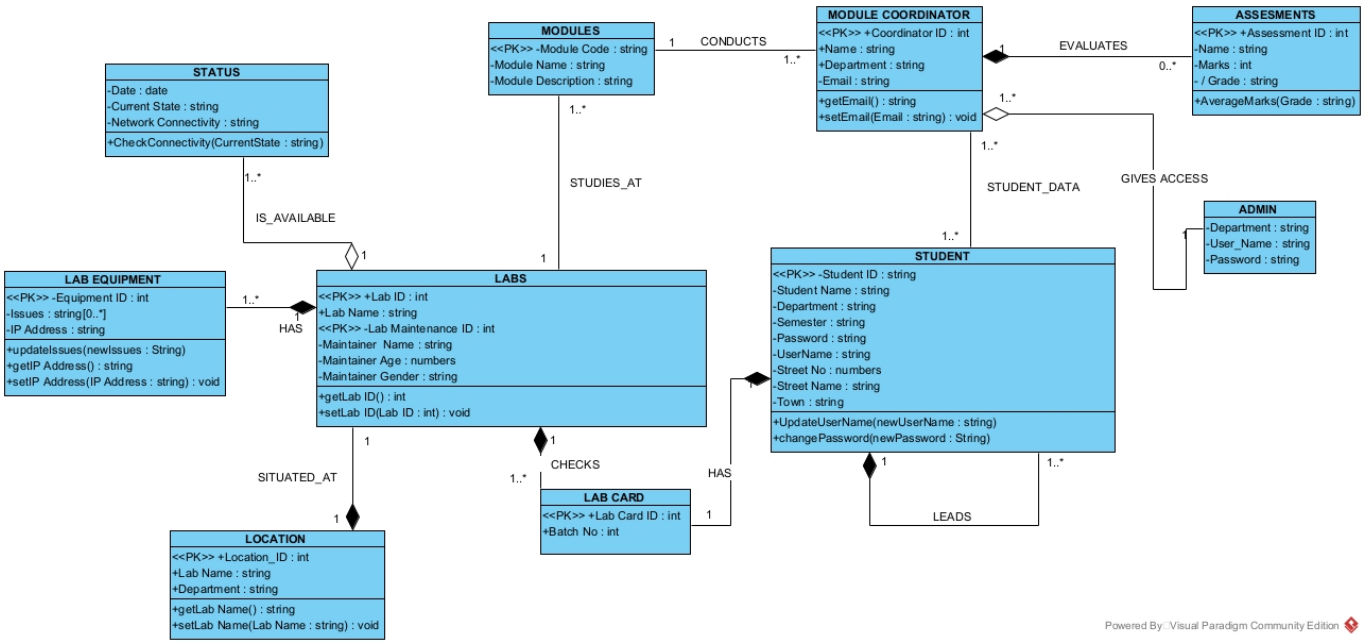


Figure 3 : UML Class Diagram for the Laboratory Management System

CHAPTER 3 – IMPLEMENTATION

Schema Creation

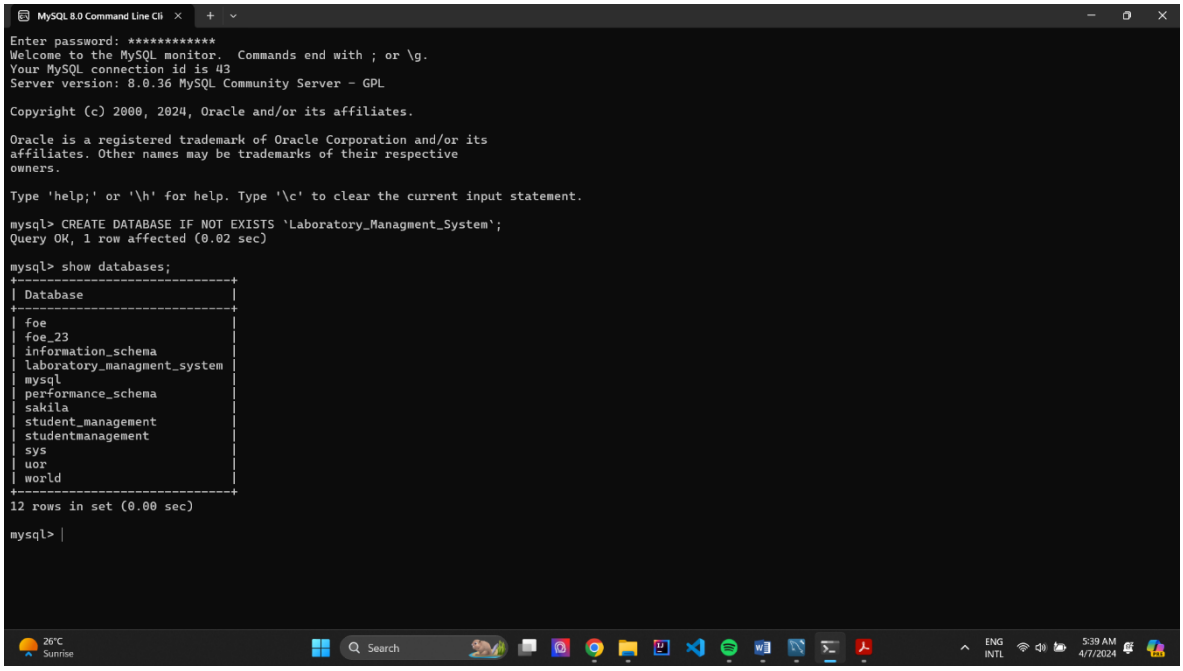


Figure 4 : Creation of Schema Laboratory Management System

Table Definitions

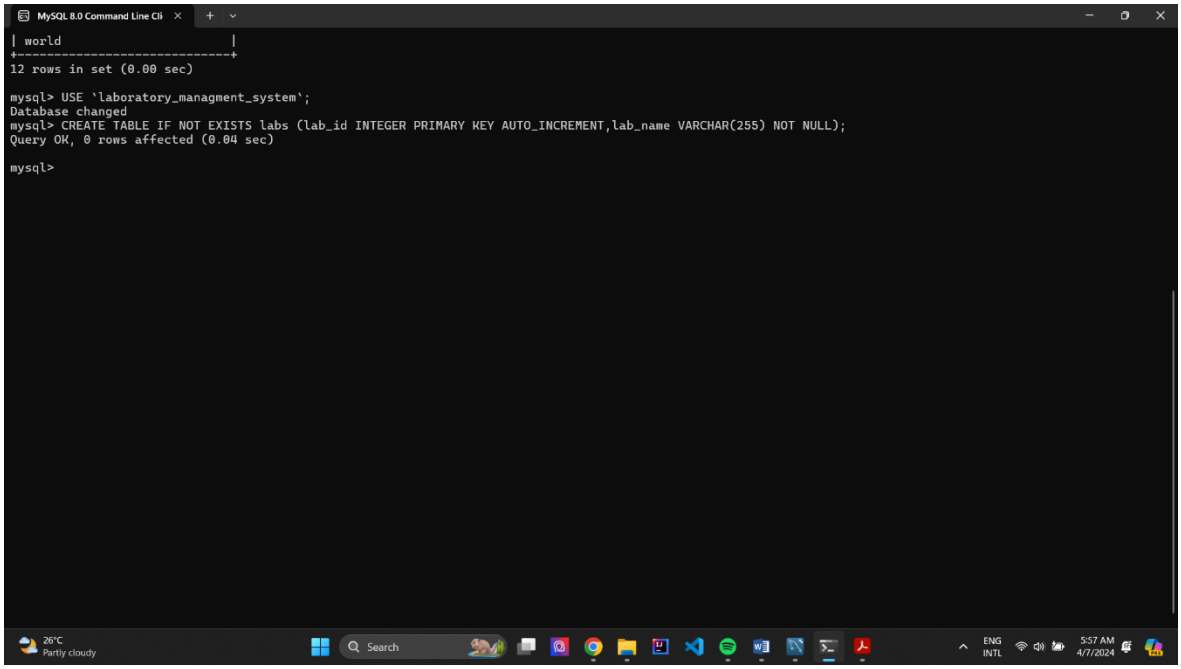


Figure 5 : Creation of Labs Table

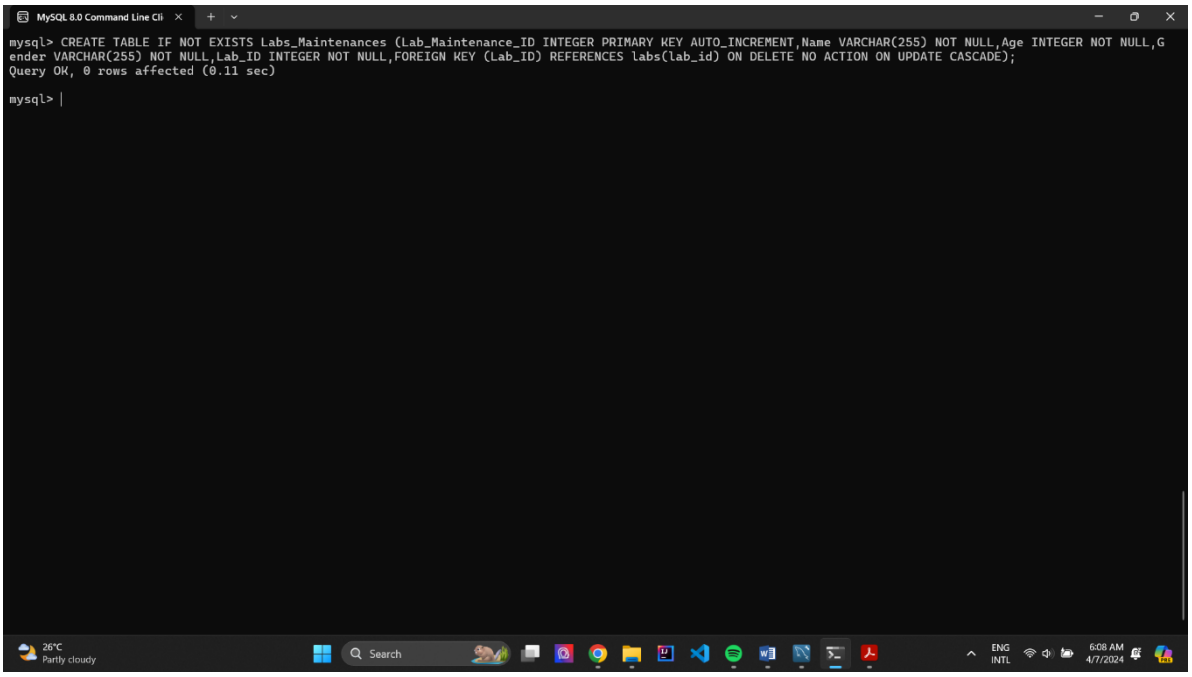


Figure 6 : Creation of Lab Maintenance Table

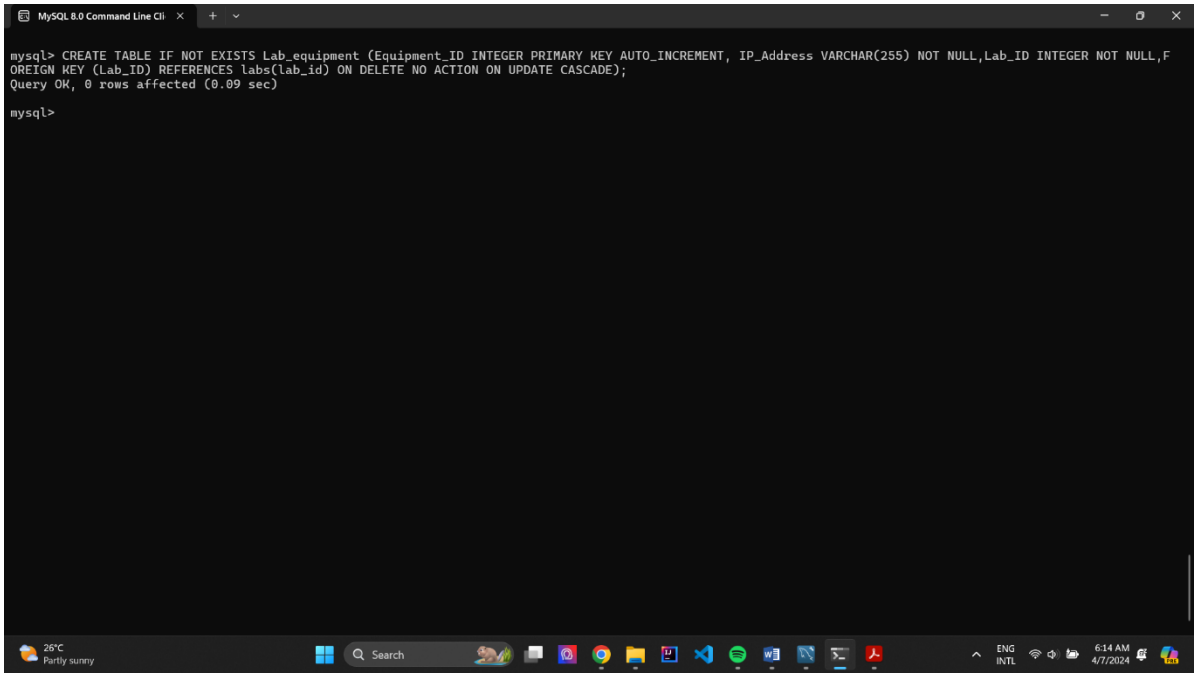


Figure 7 : Creation of Lab Equipment Table

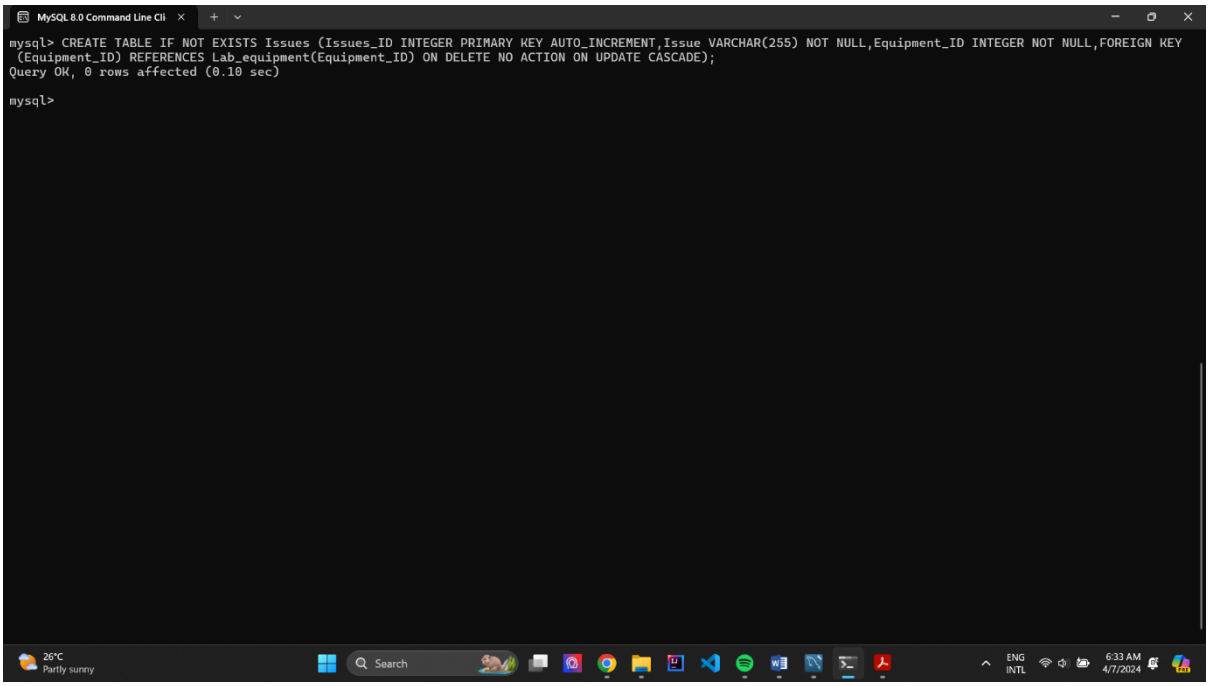


Figure 8 : Creation of Issues Table

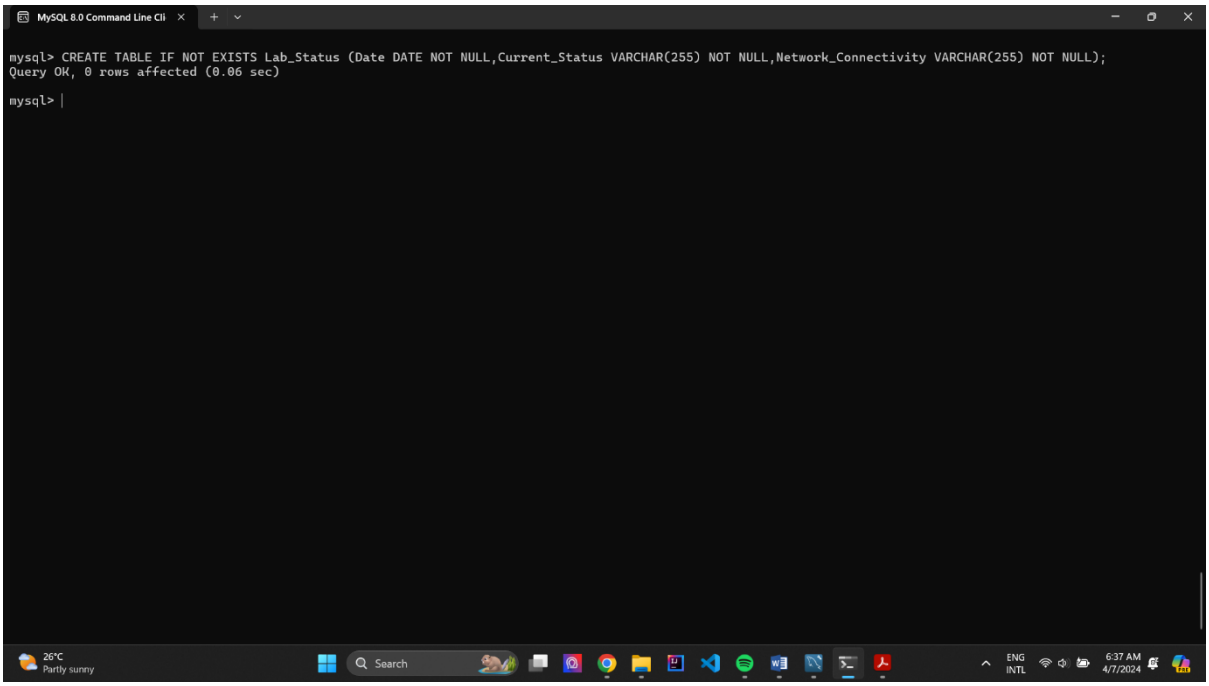


Figure 9 : Creation of Lab Status Table

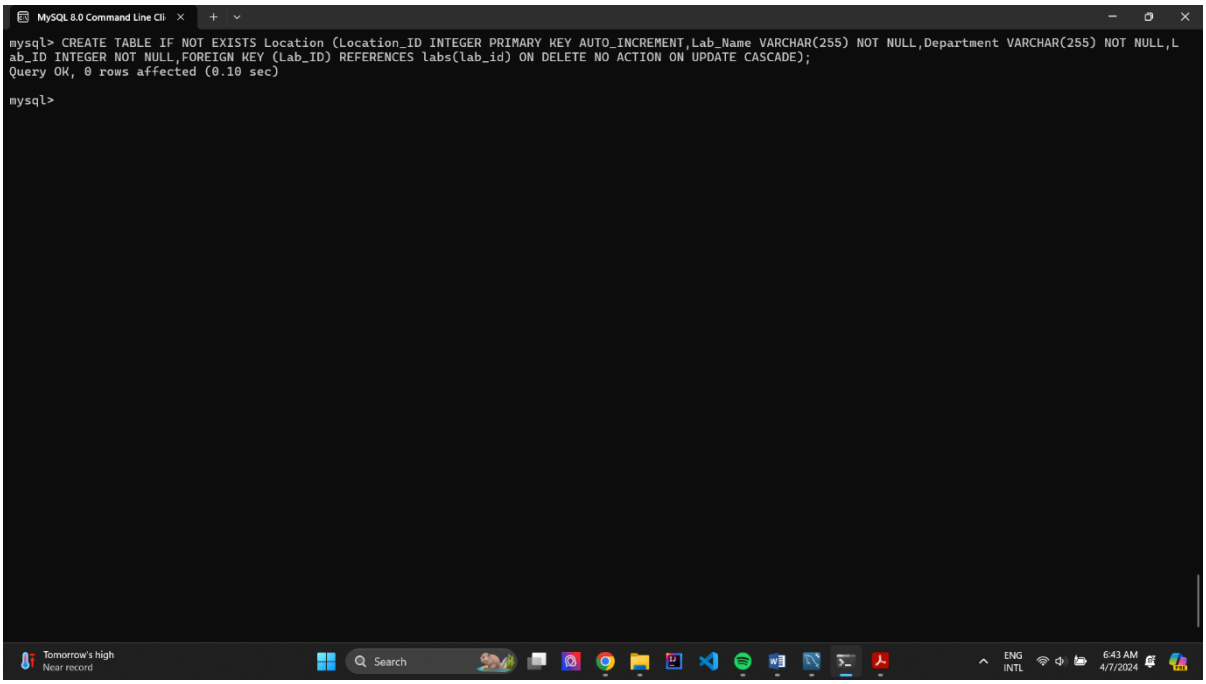


Figure 10 : Creation of Location Table

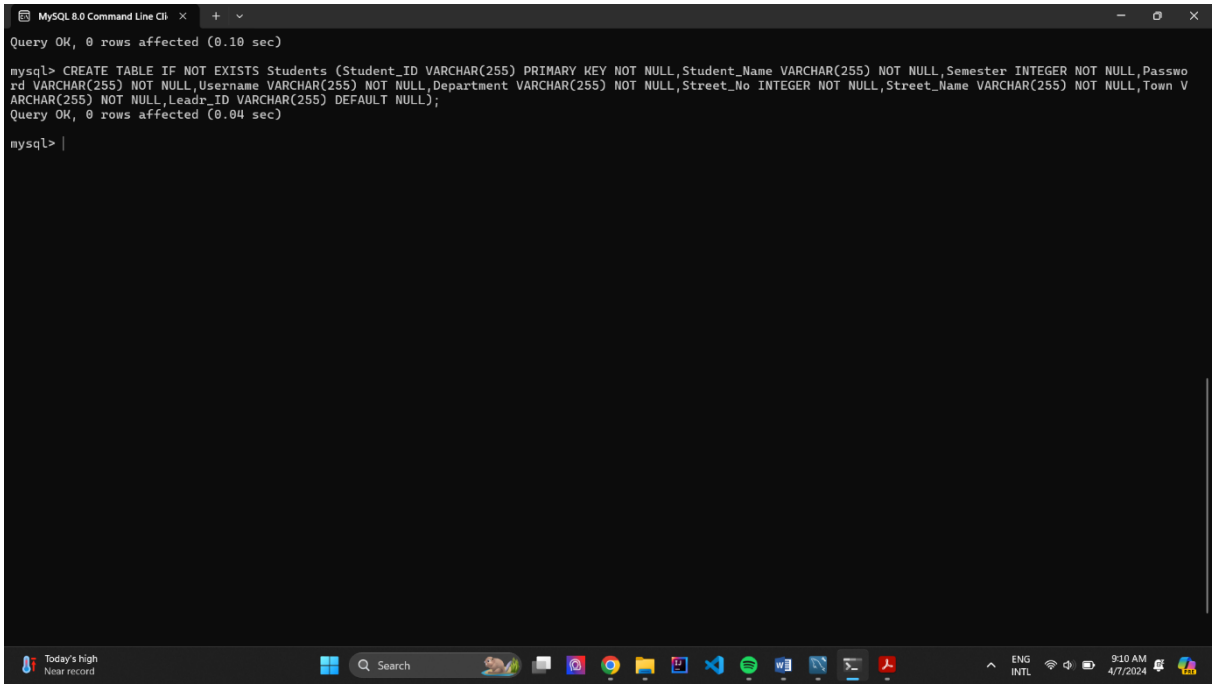


Figure 11 : Creation of Students Table

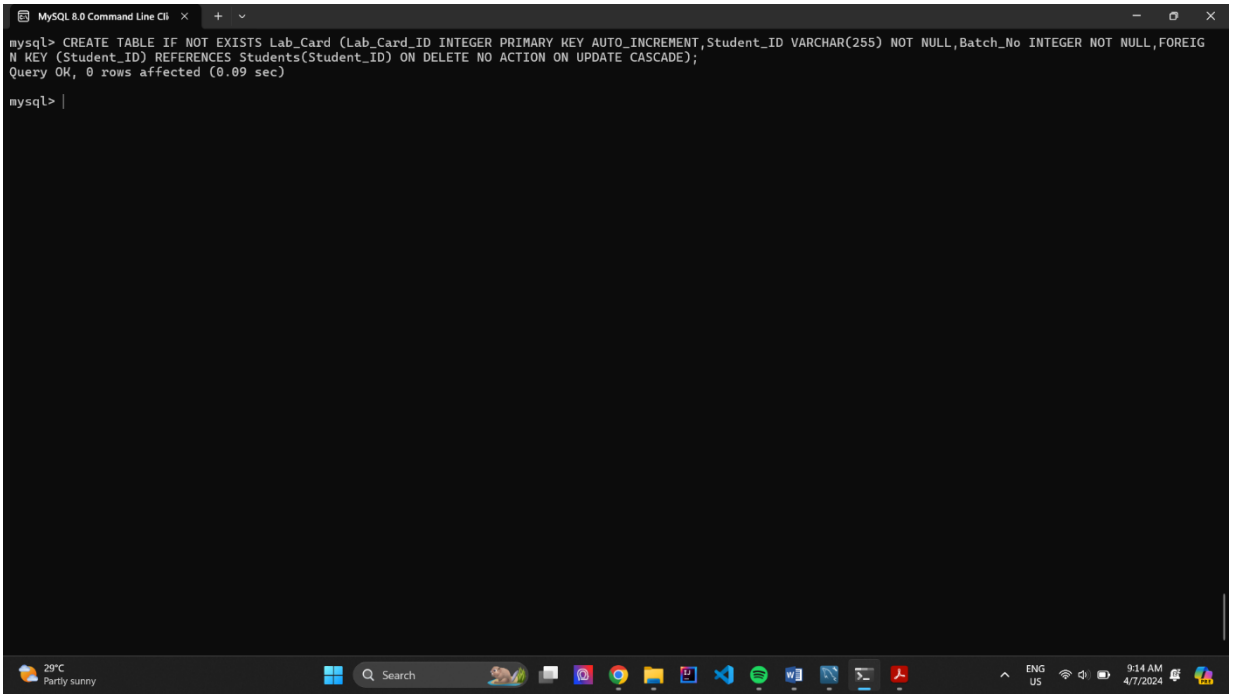


Figure 12 : Creation of Lab Card Table

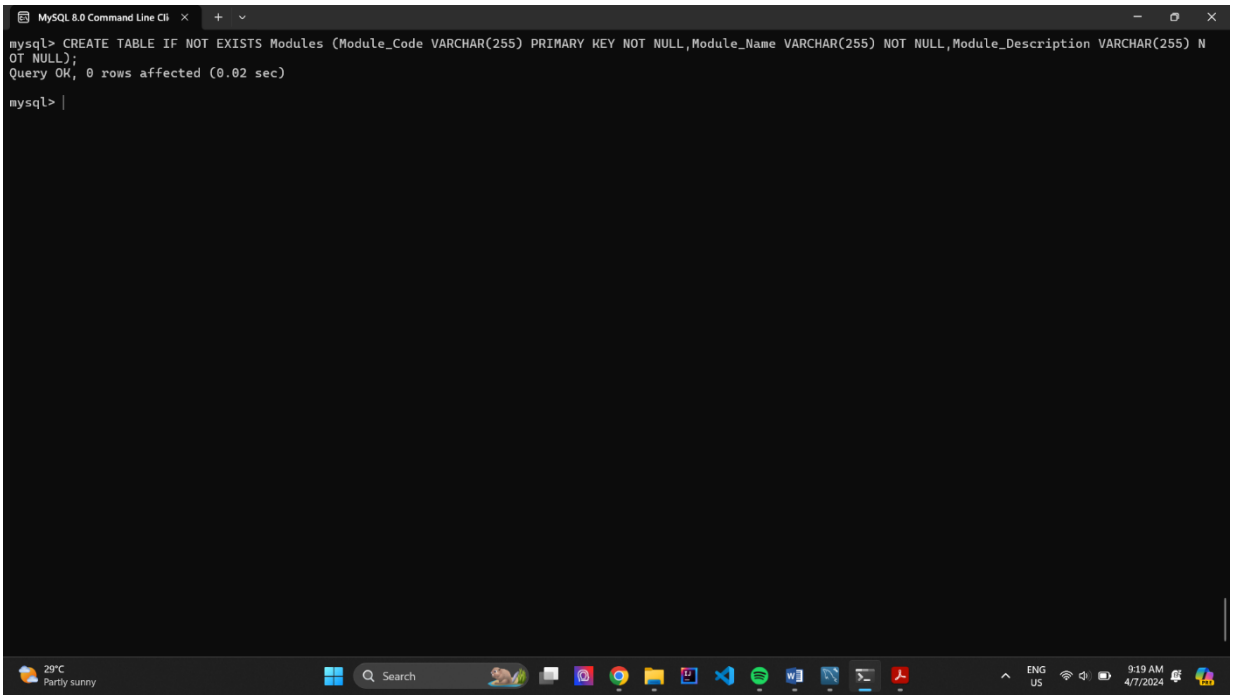


Figure 13 : Creation of Modules Table

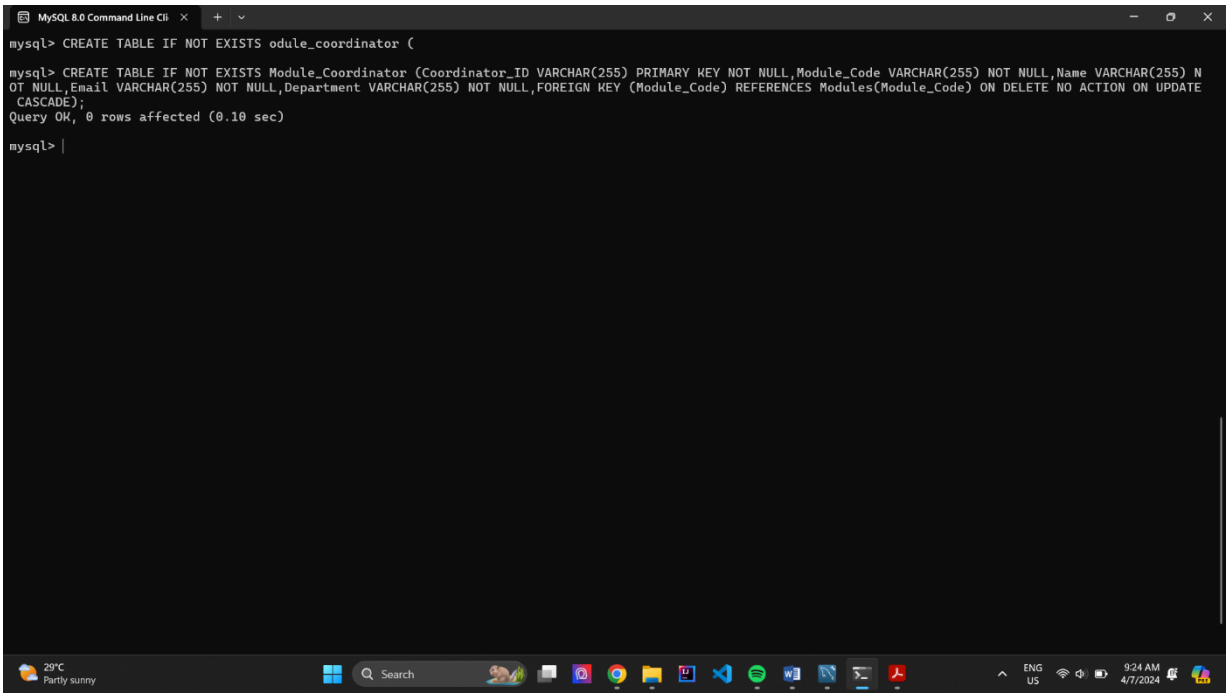


Figure 14 : Creation of Module Coordinator Table

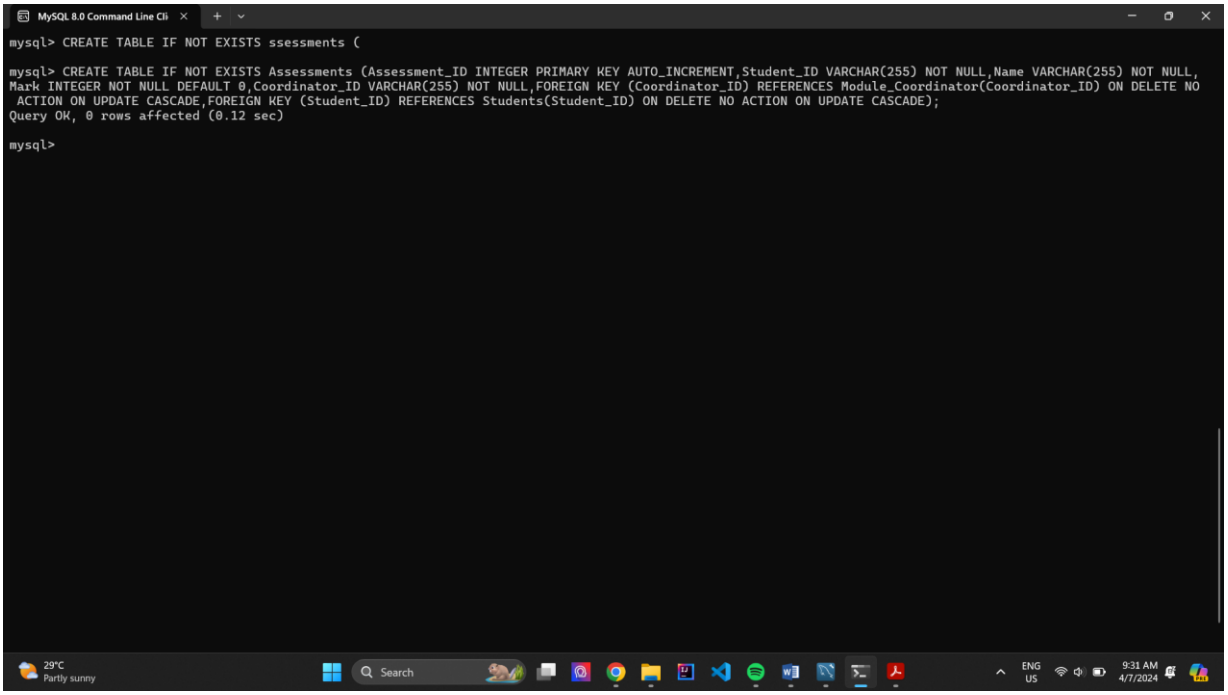


Figure 15 : Creation of Assessments Table

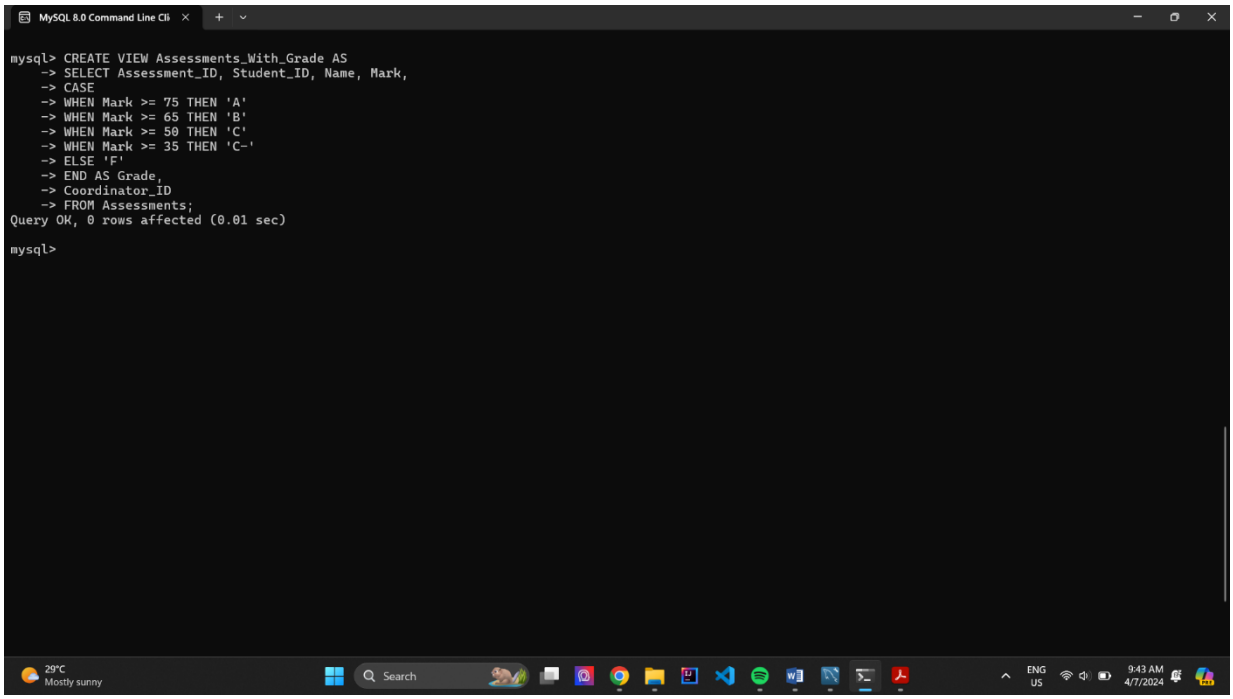


Figure 16 : Definition of the Derived attribute 'Grade'

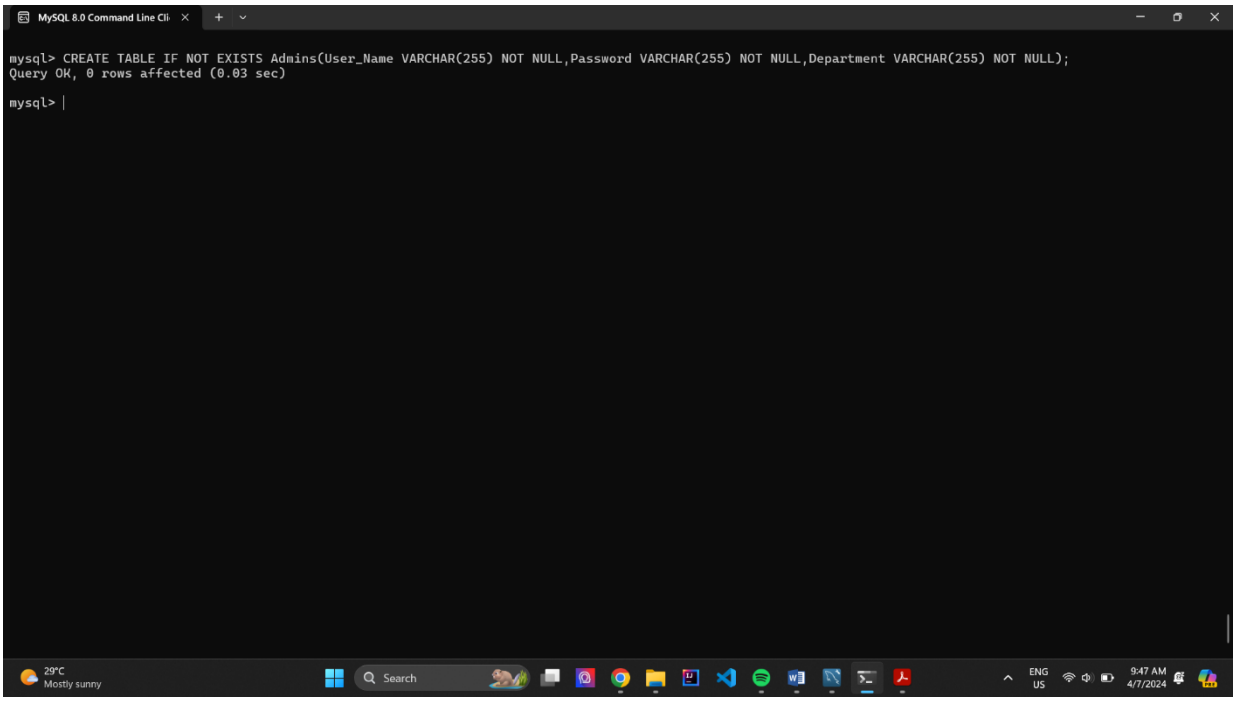


Figure 17 : Creation of Admins Table

Insertion

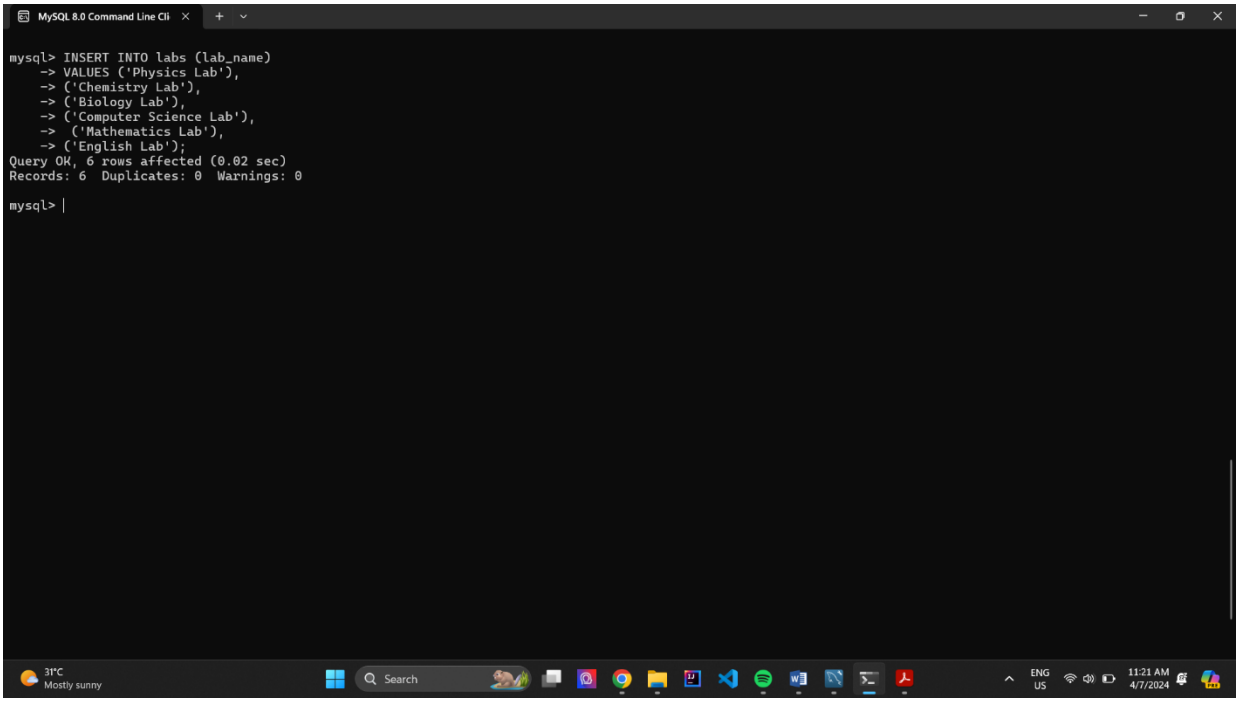


Figure 18 : Data Insertion to the Labs Table

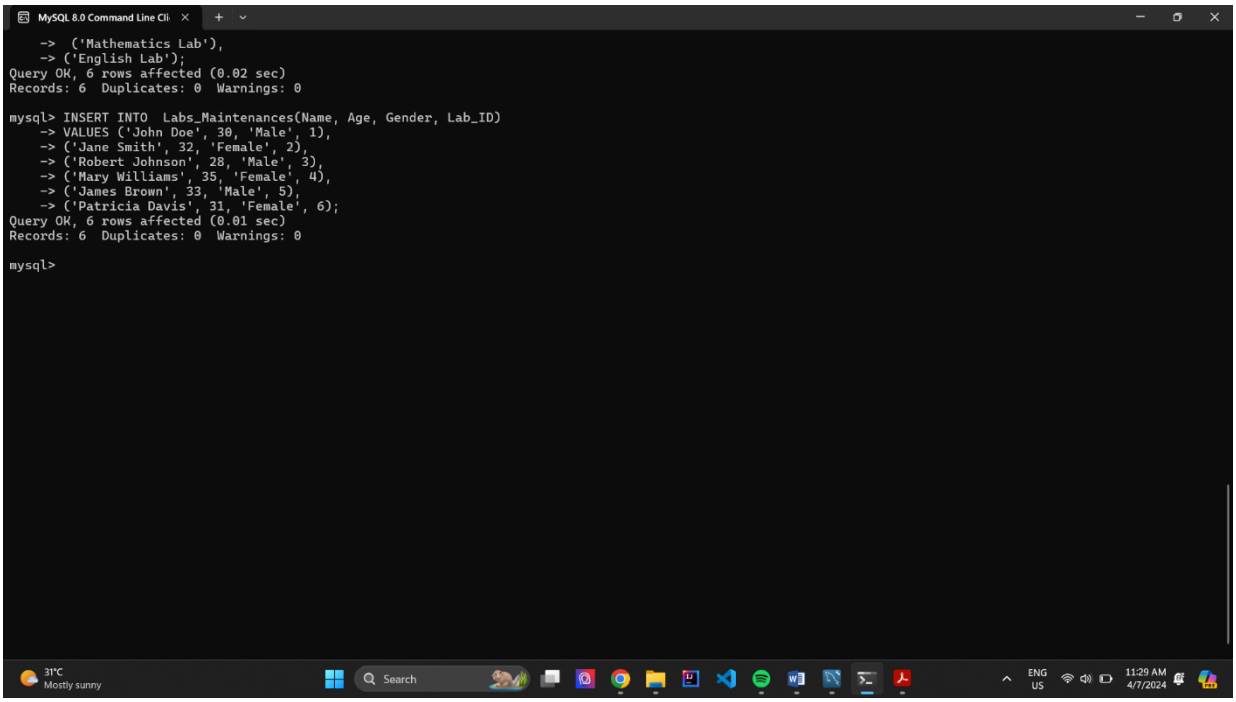


Figure 19 : Data Insertion to the Lab Maintenance Table

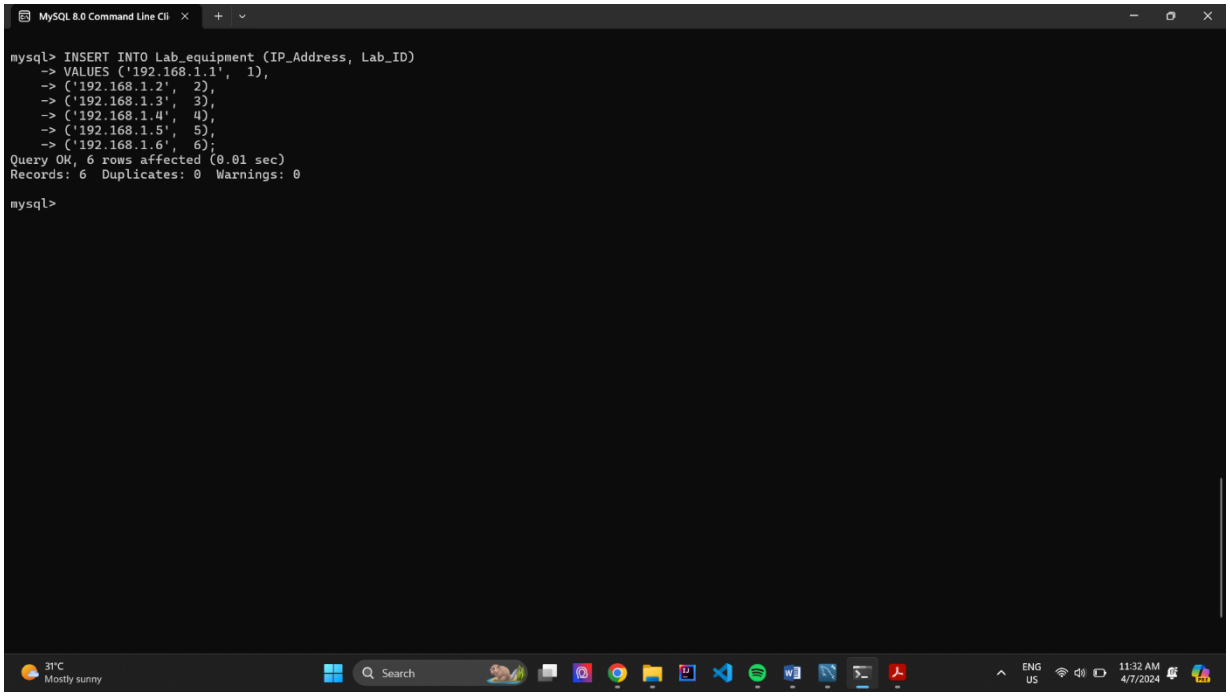


Figure 20 : Data Insertion to the Lab Equipment Table

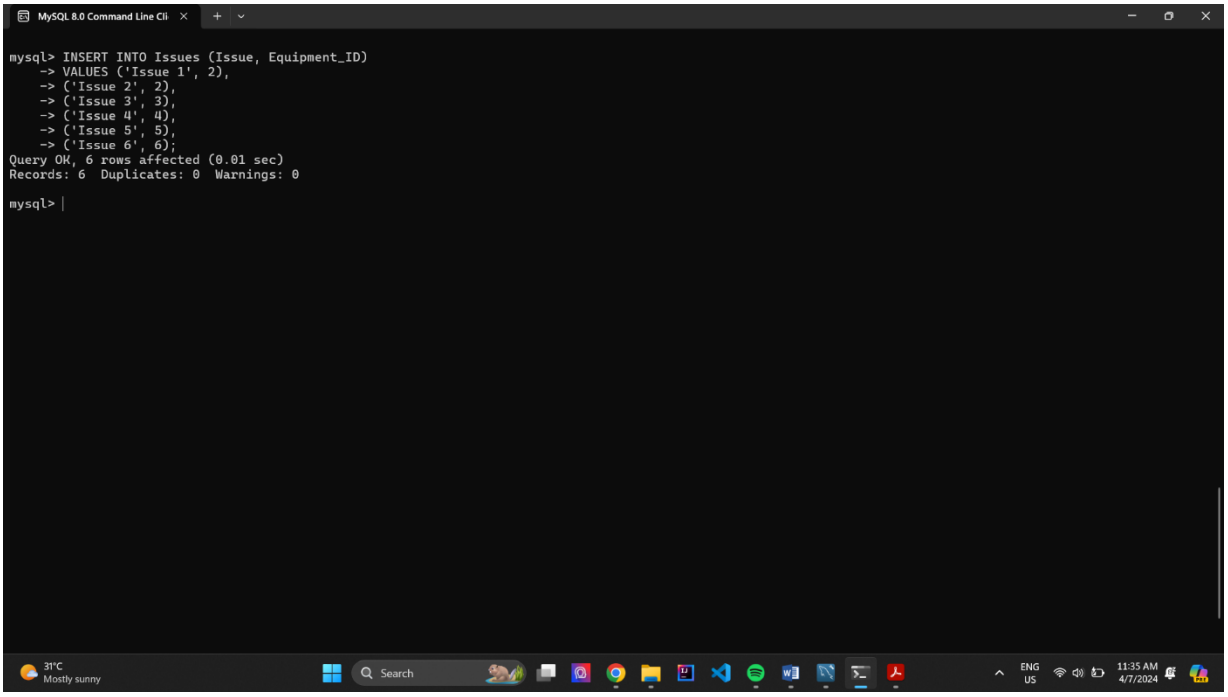


Figure 21 : Data Insertion to the Issues Table

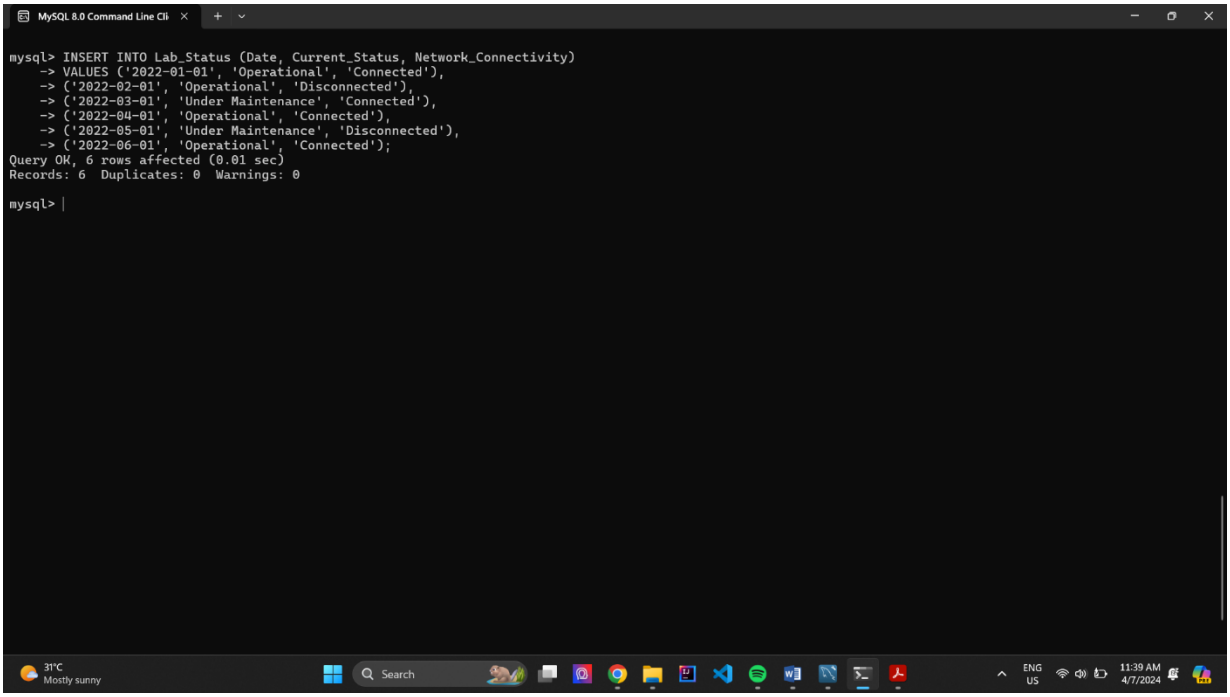


Figure 22 : Data Insertion to the Lab Status Table

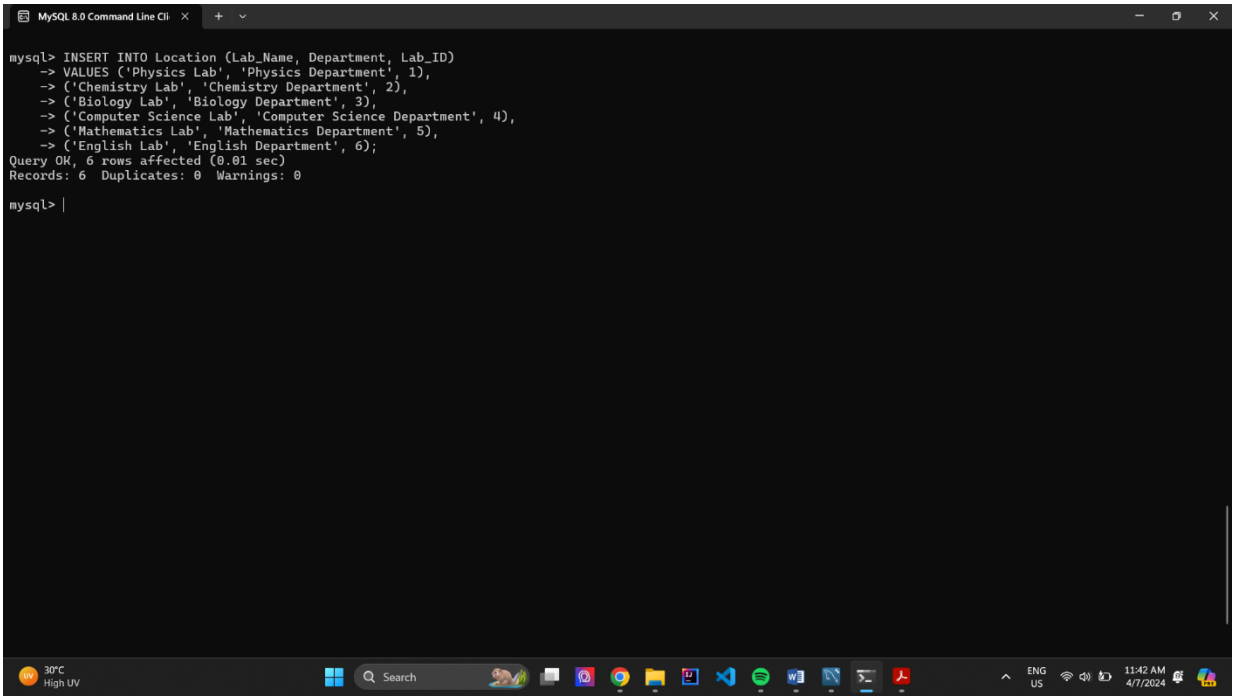


Figure 23 : Data Insertion to the Location Table

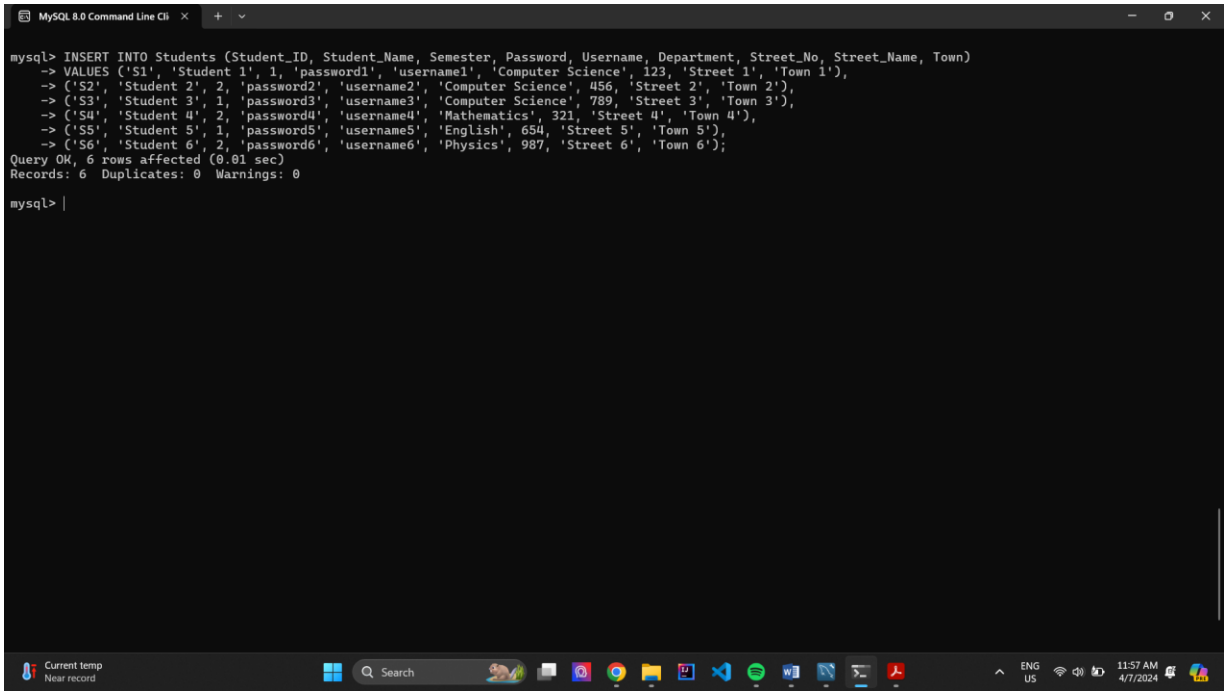


Figure 24 : Data Insertion to the Students Table

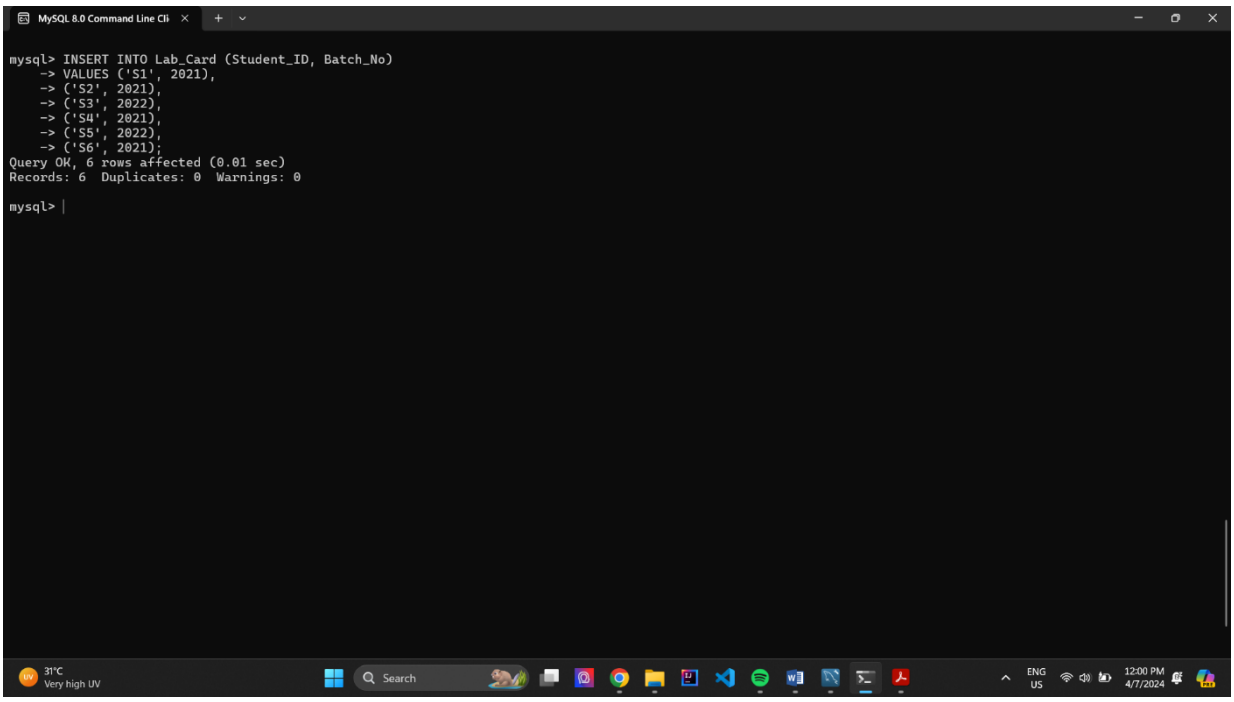


Figure 25 : Data Insertion to the Lab Card Table

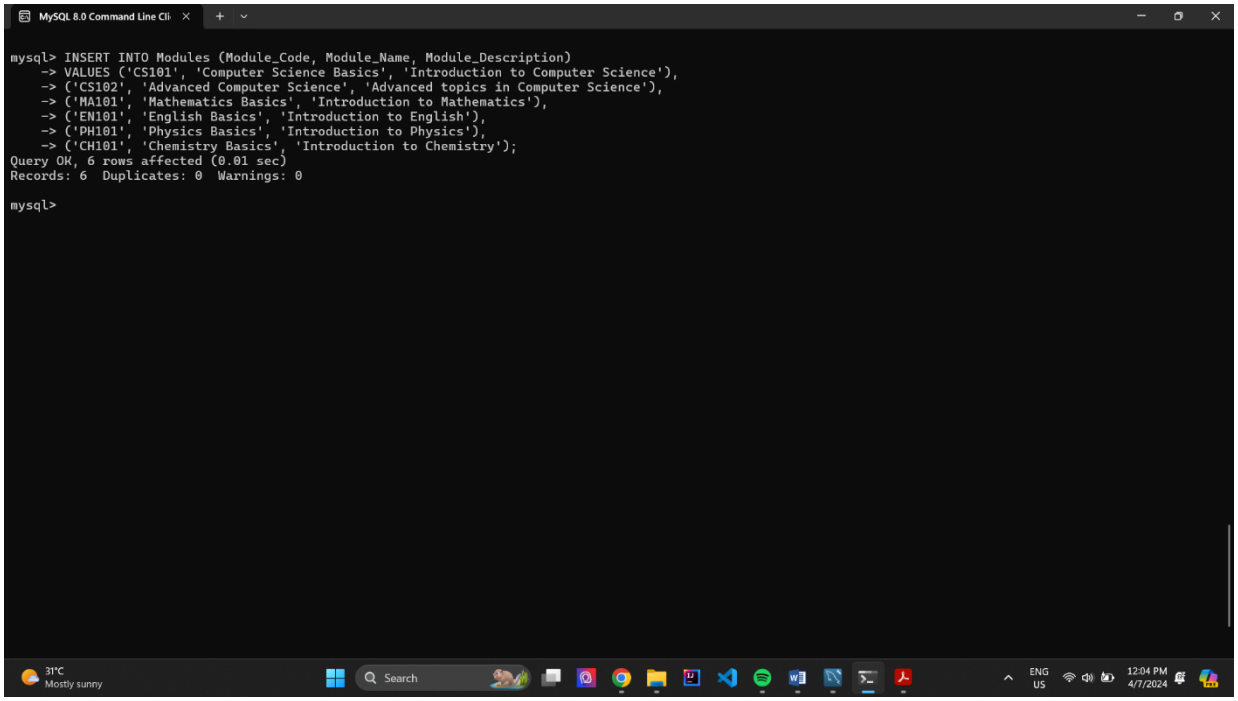


Figure 26 : Data Insertion to the Modules Table

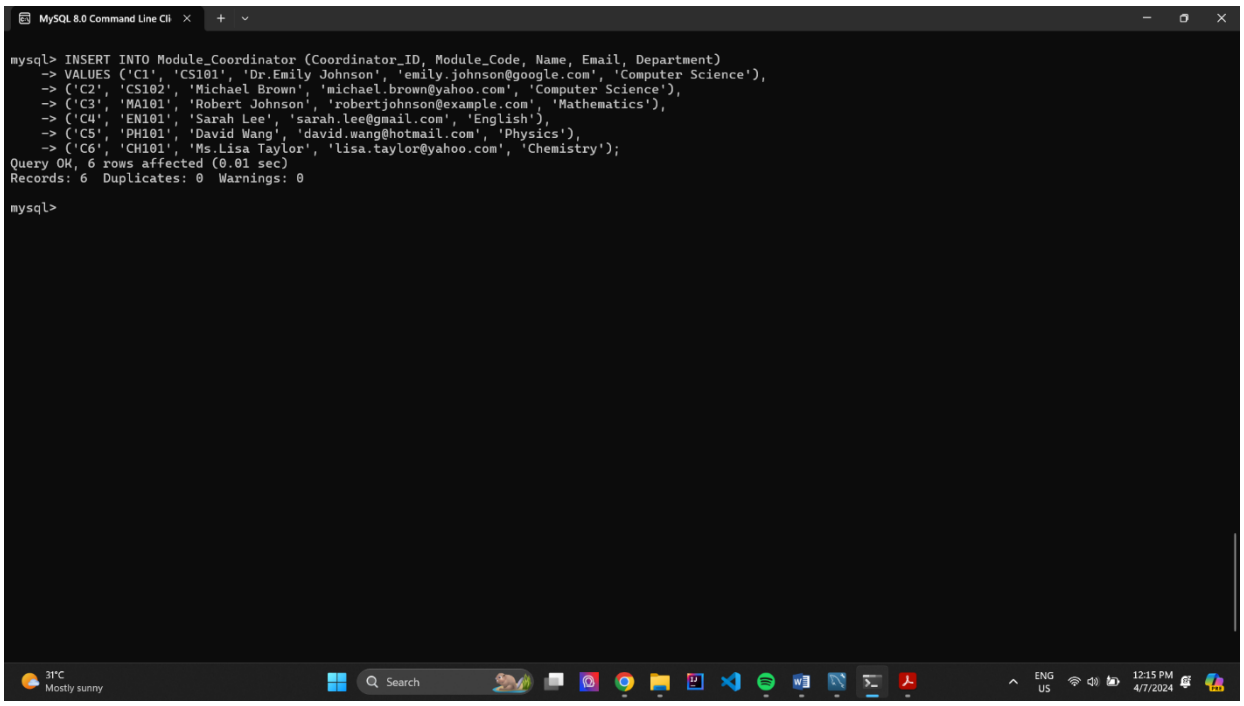


Figure 27 : Data Insertion to the Module Coordinator Table

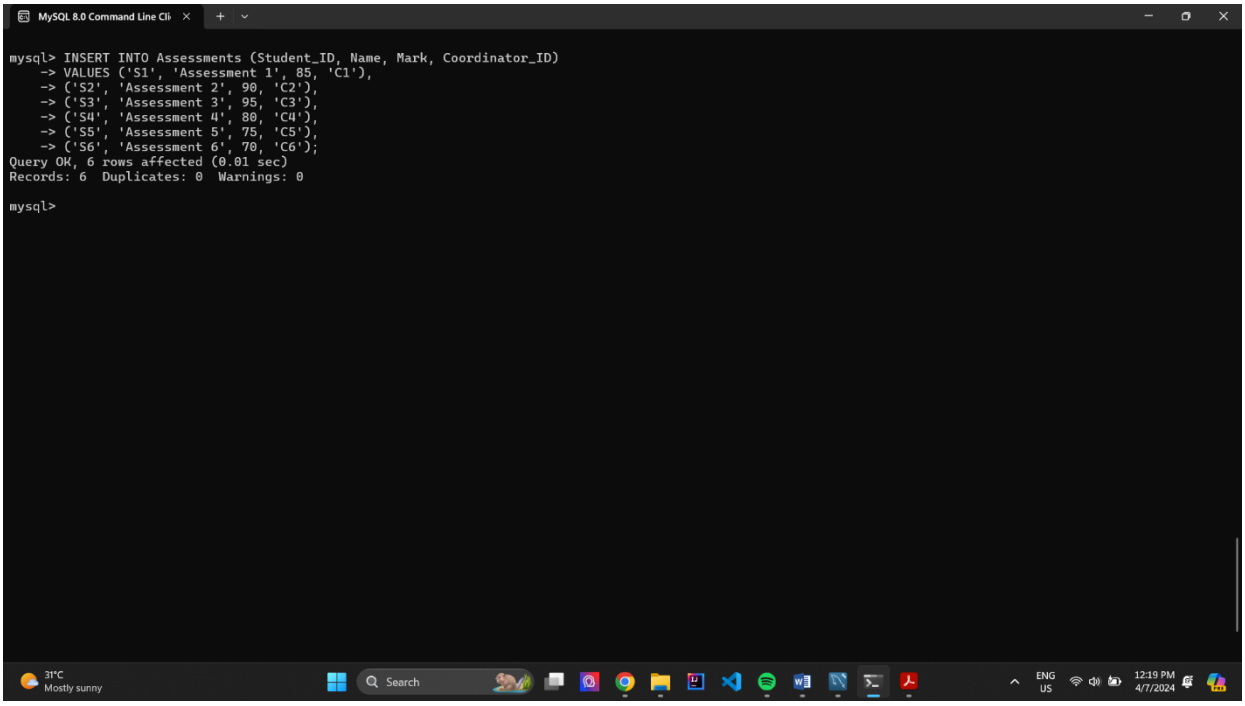


Figure 28 : Data Insertion to the Assessments Table

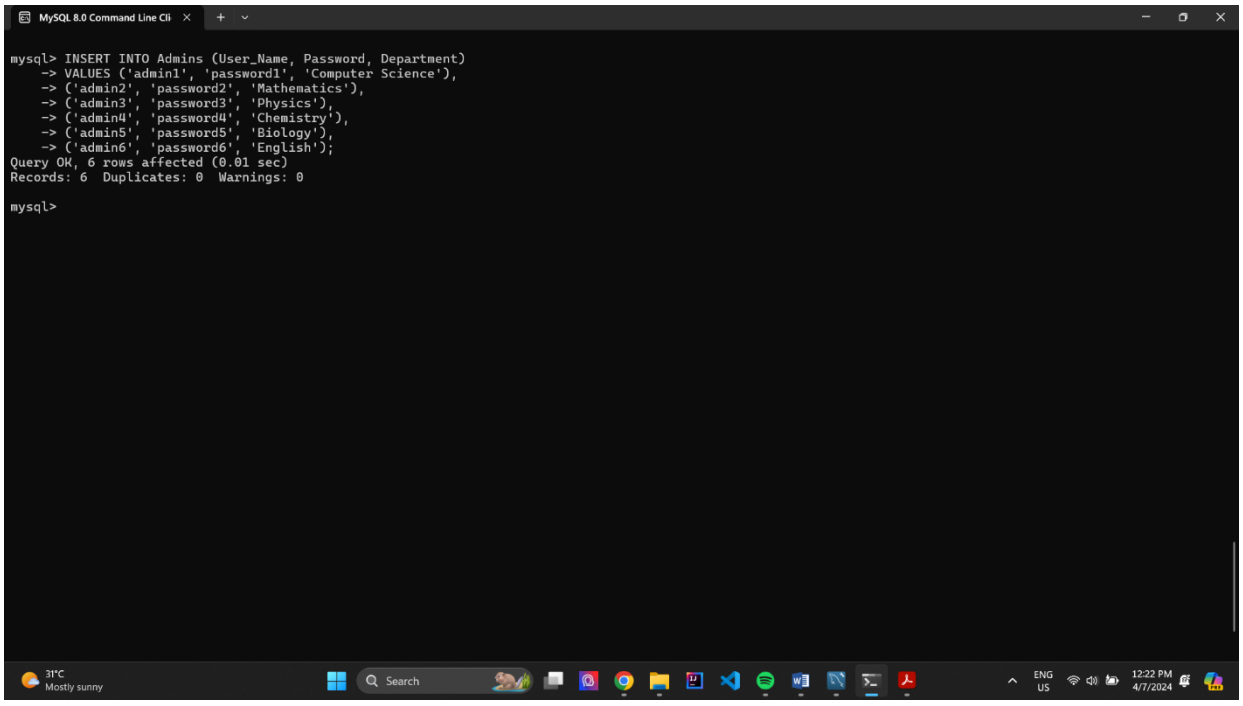


Figure 29 : Data Insertion to the Admins Table

Update Operation

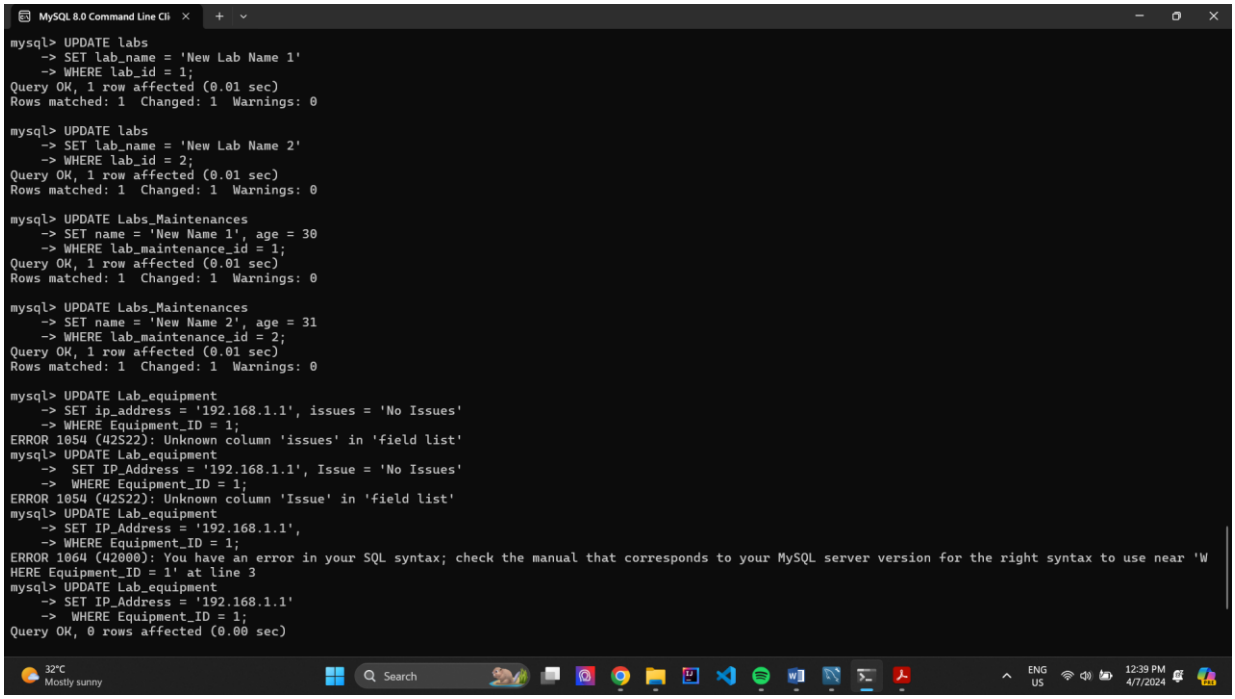


Figure 30 : Data Updates in Labs and Lab Maintenance Tables

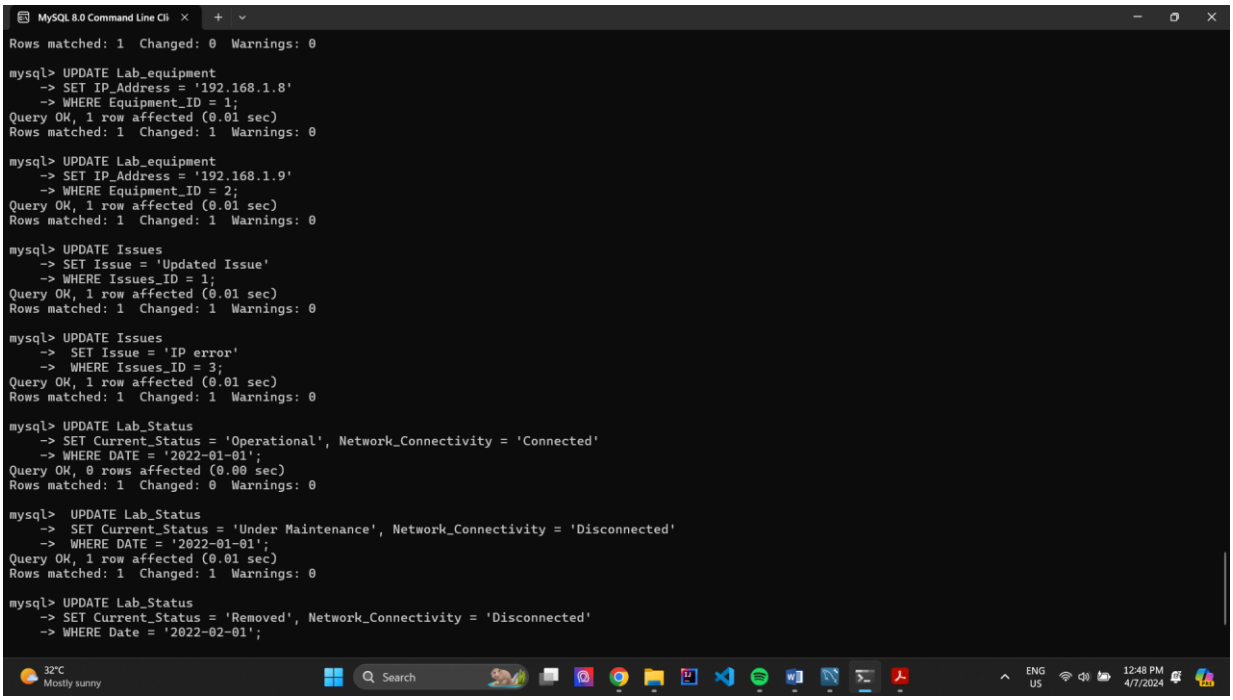


Figure 31 : Data Updates in Lab Equipment and Issues Tables

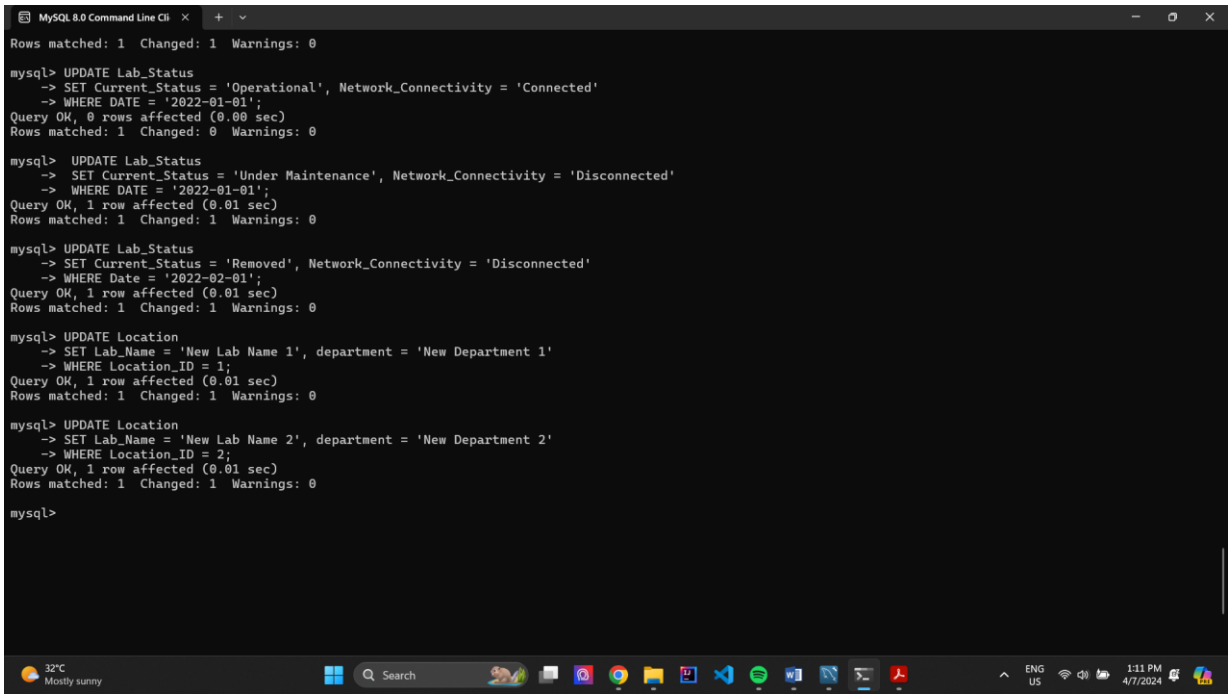


Figure 32 : Data Updates in Lab Status and Location Tables

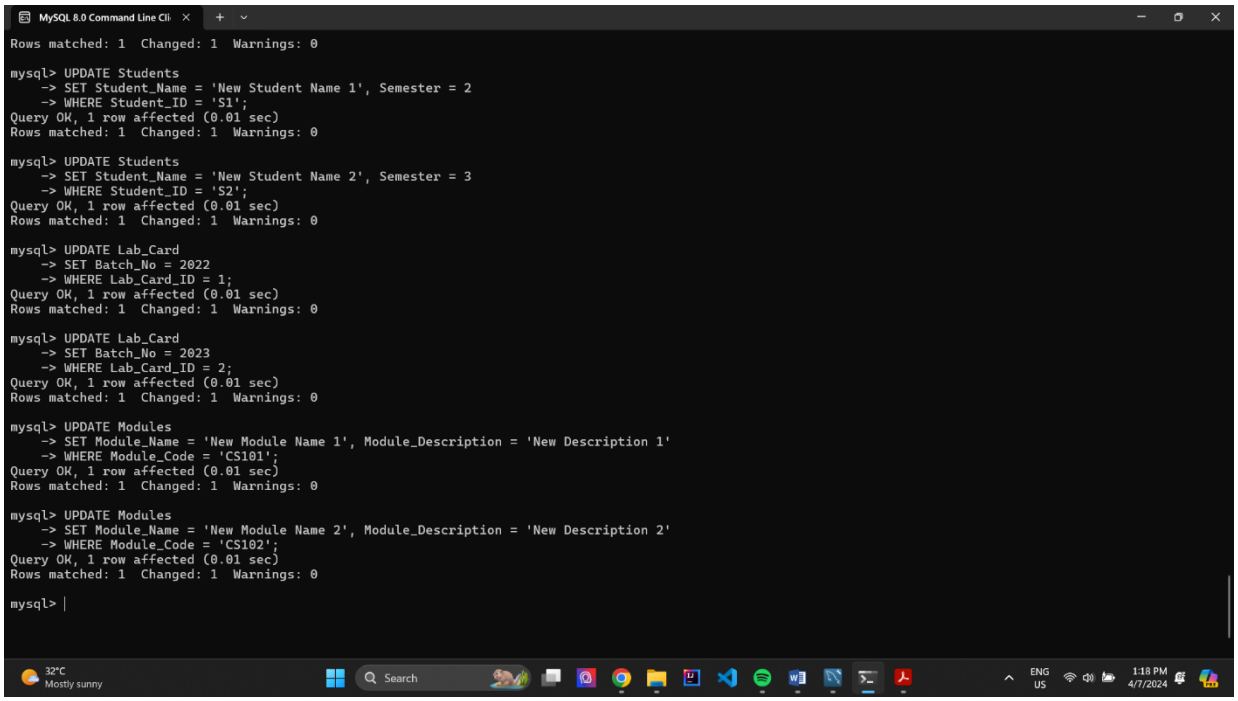


Figure 33 : Data Updates in Students, Lab Card and Modules Tables

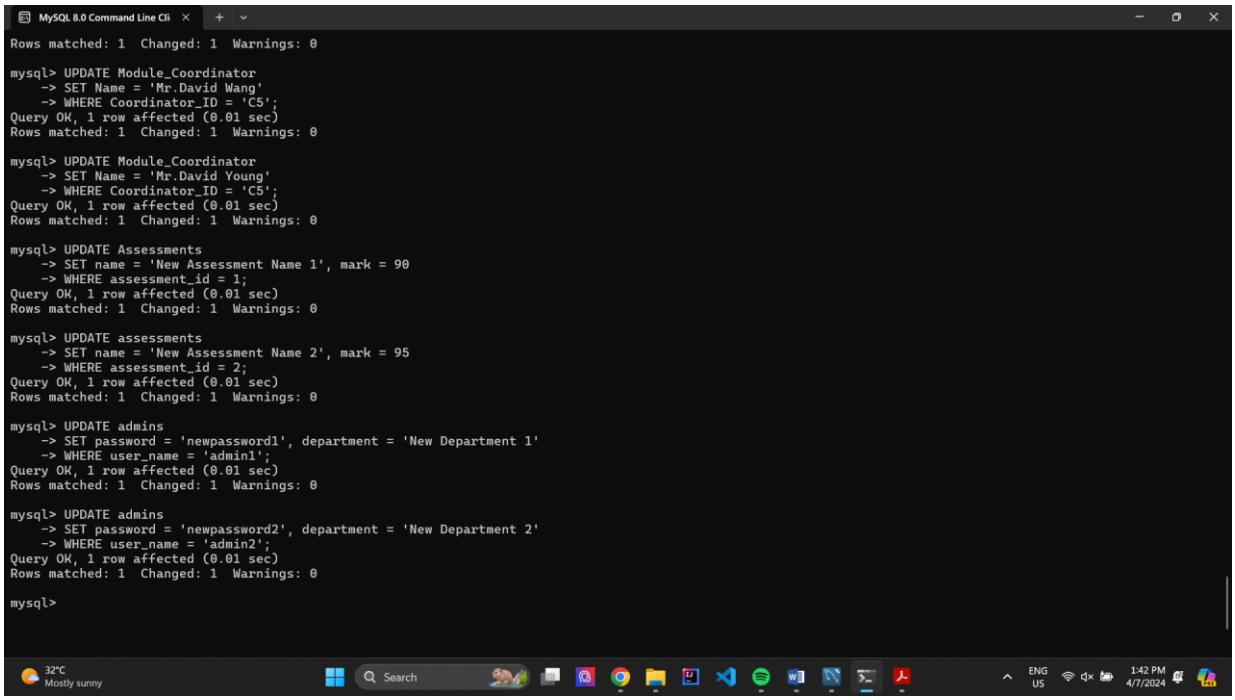


Figure 34 : Data Updates in Module Coordinator, Assessments and Admins Tables

Deletion

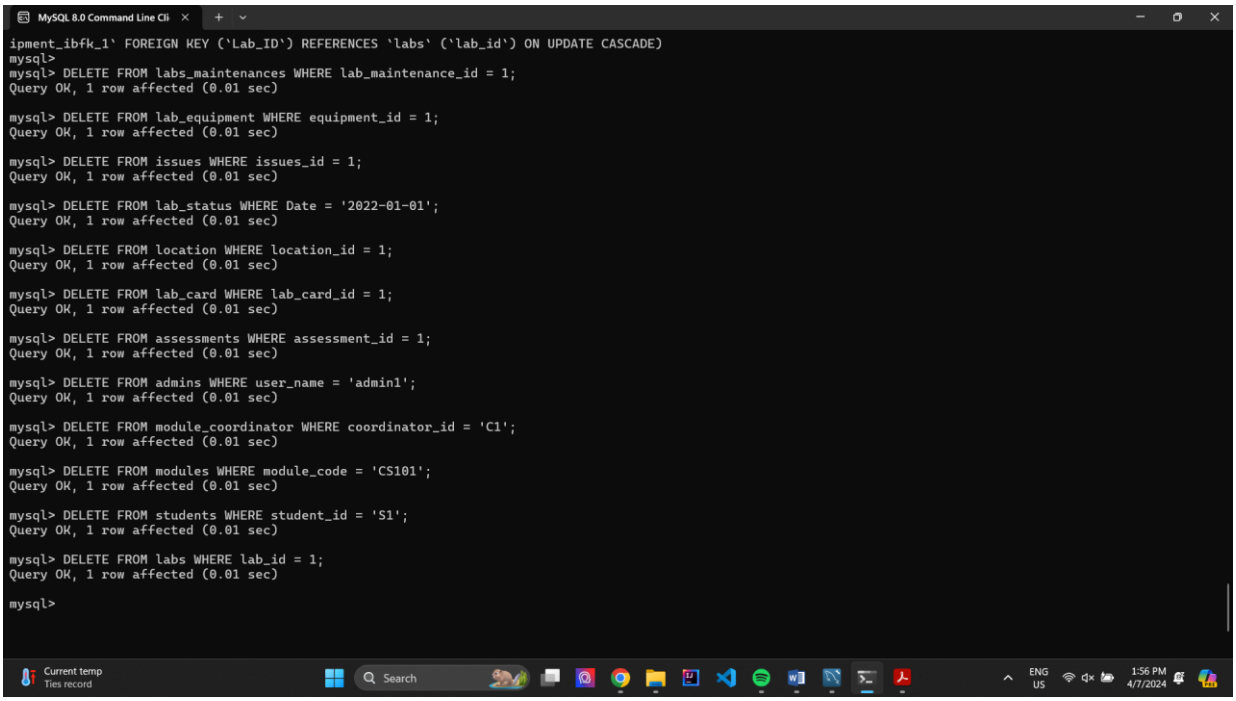


Figure 35 : Deleting data from each table

CHAPTER 4 – TRANSACTION

Simple Queries

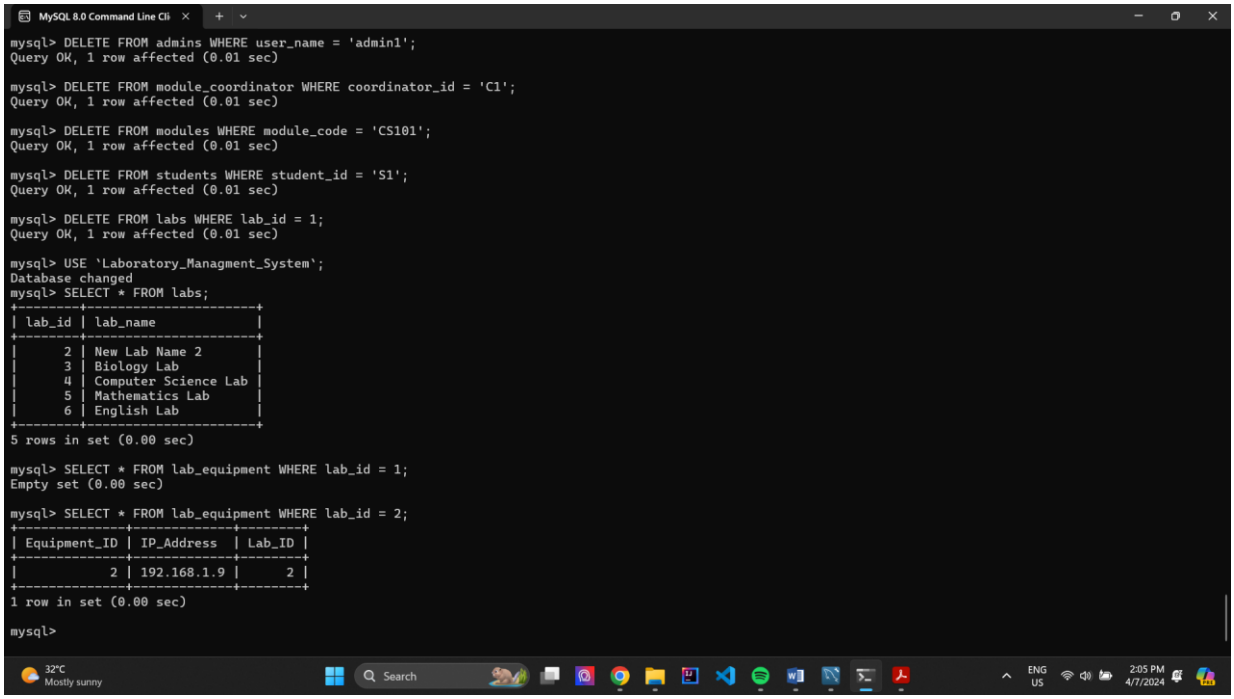


Figure 37 : Select Operation

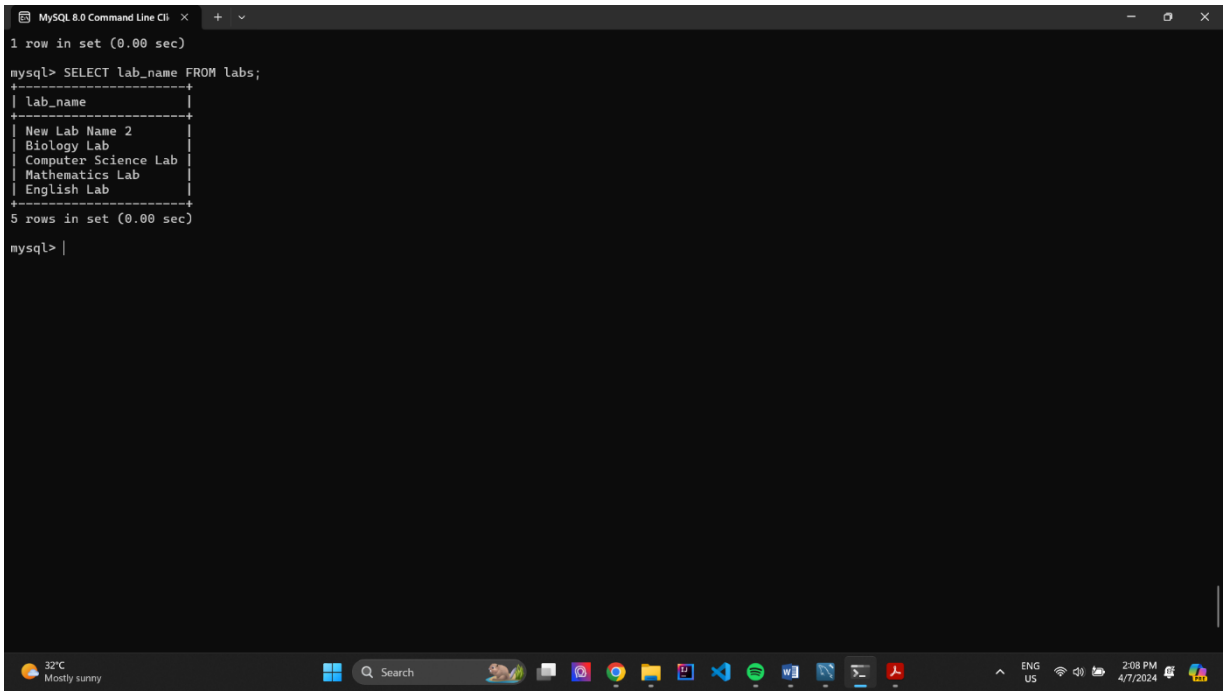


Figure 36 : Project Operation

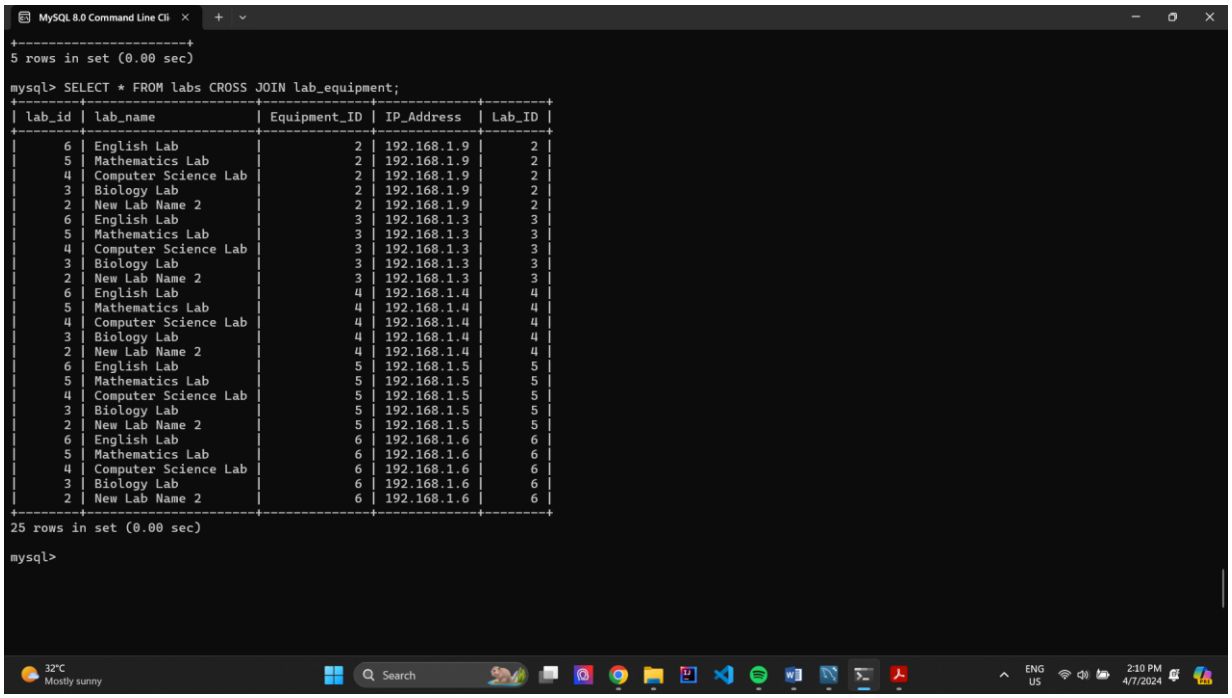


Figure 38 : Cross Product Operation

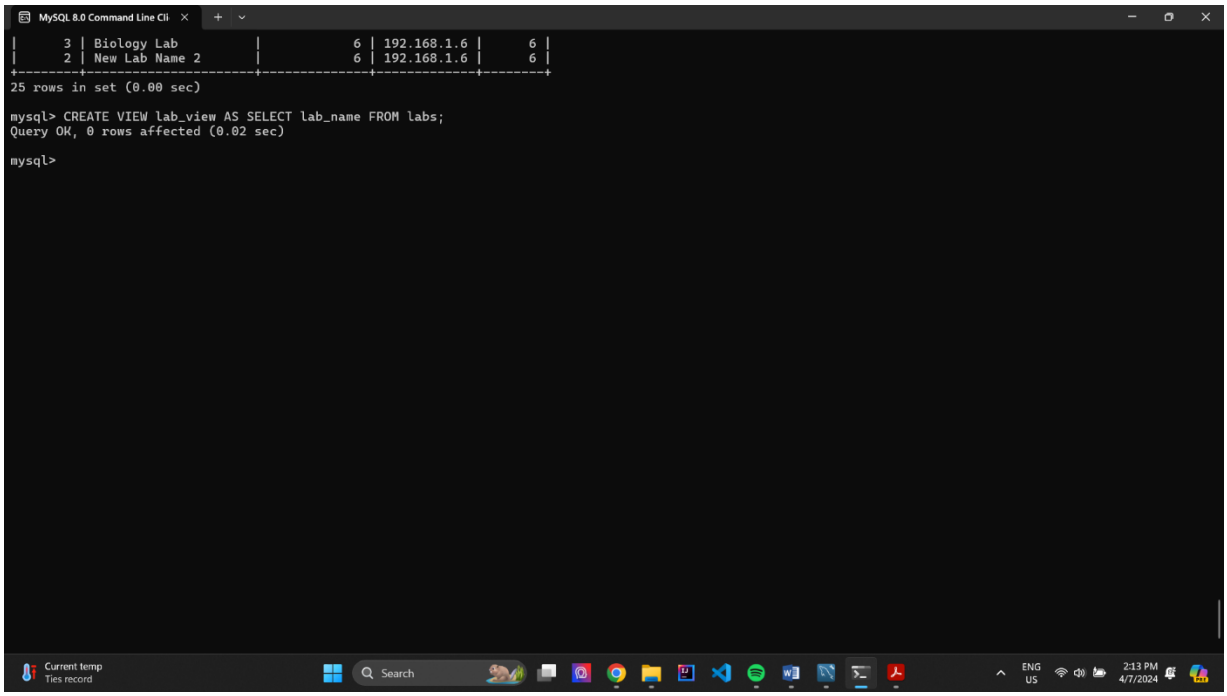


Figure 39 : User view created for the Labs Table

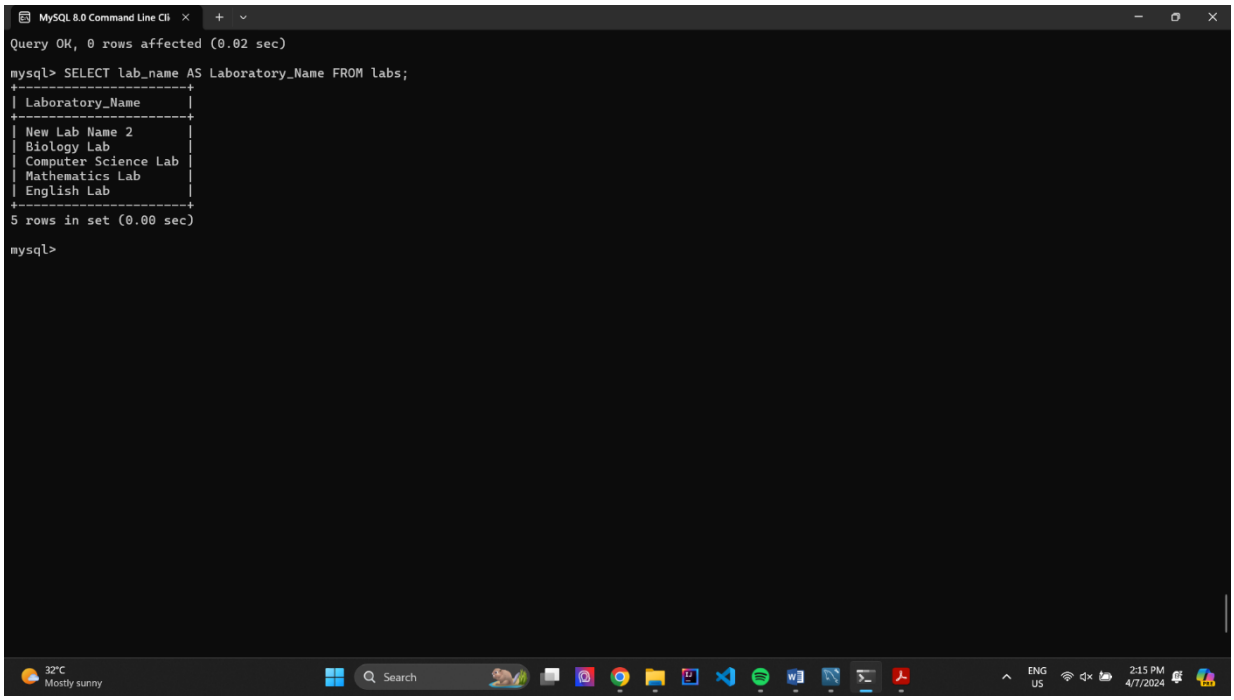


Figure 40 : Renaming Operation

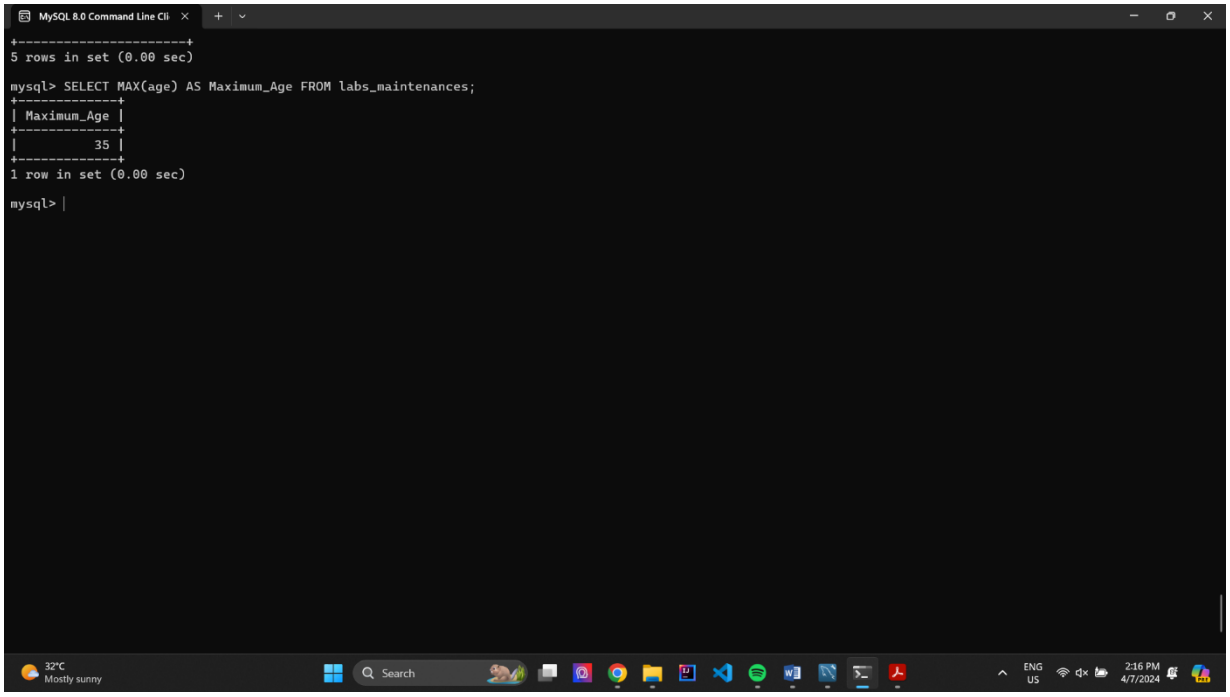


Figure 41 : Find the Maximum Age from Lab Maintenance Table (Aggregation Functions)

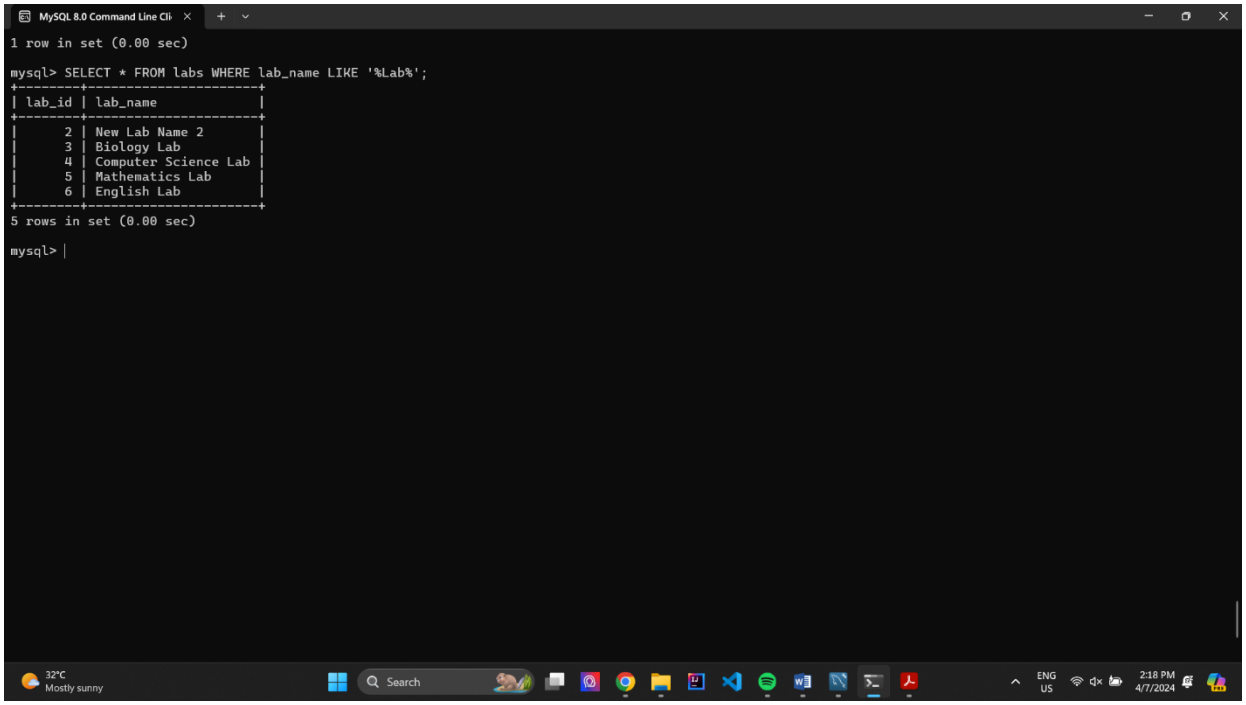


Figure 42 : Use of the keyword 'Like'

Complex Queries

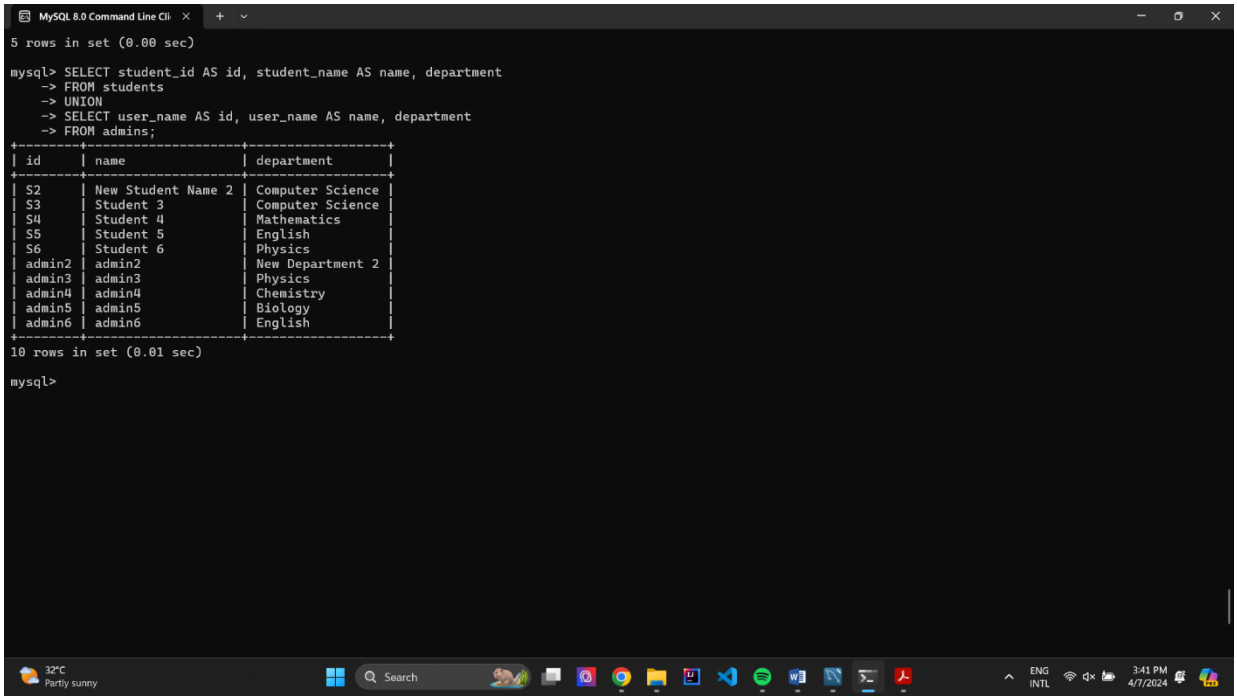


Figure 43 : Union Operation

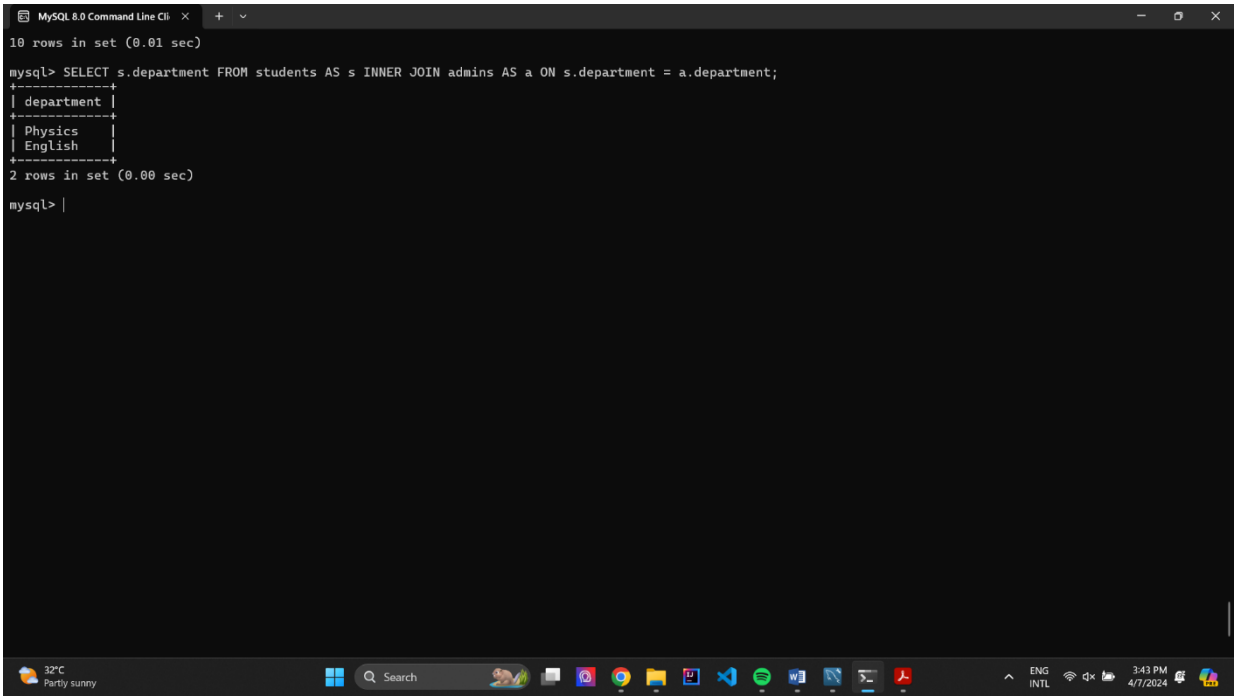


Figure 44 : Intersection Operation

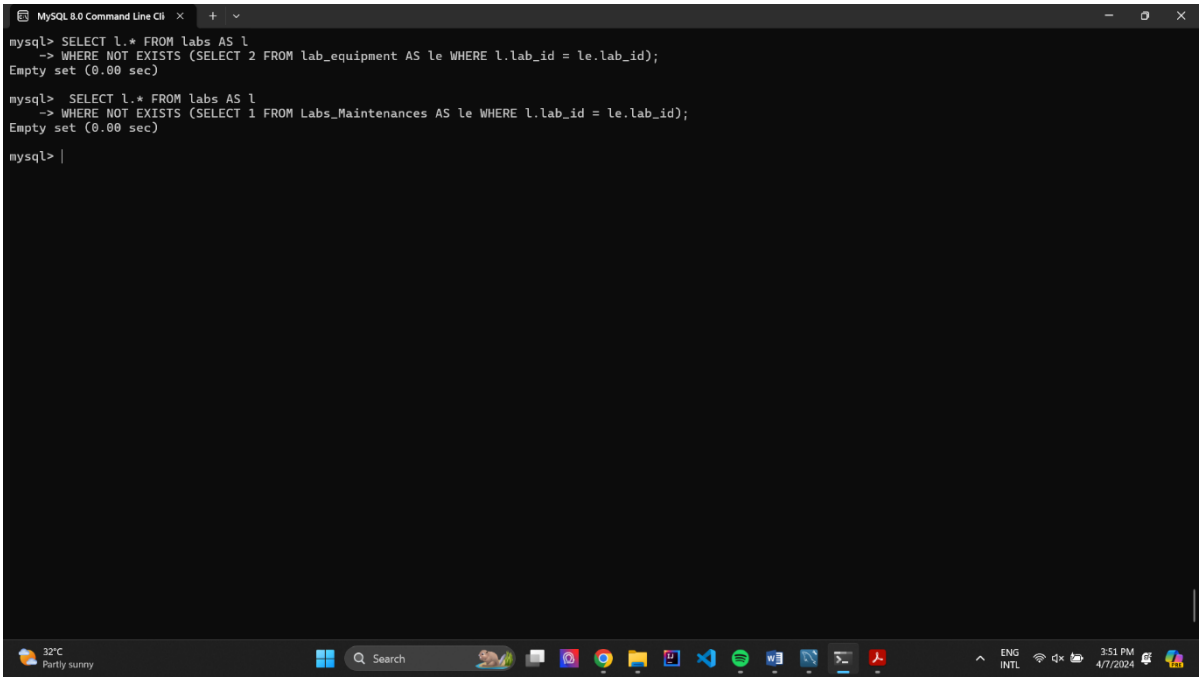


Figure 45 : Set Difference Operation

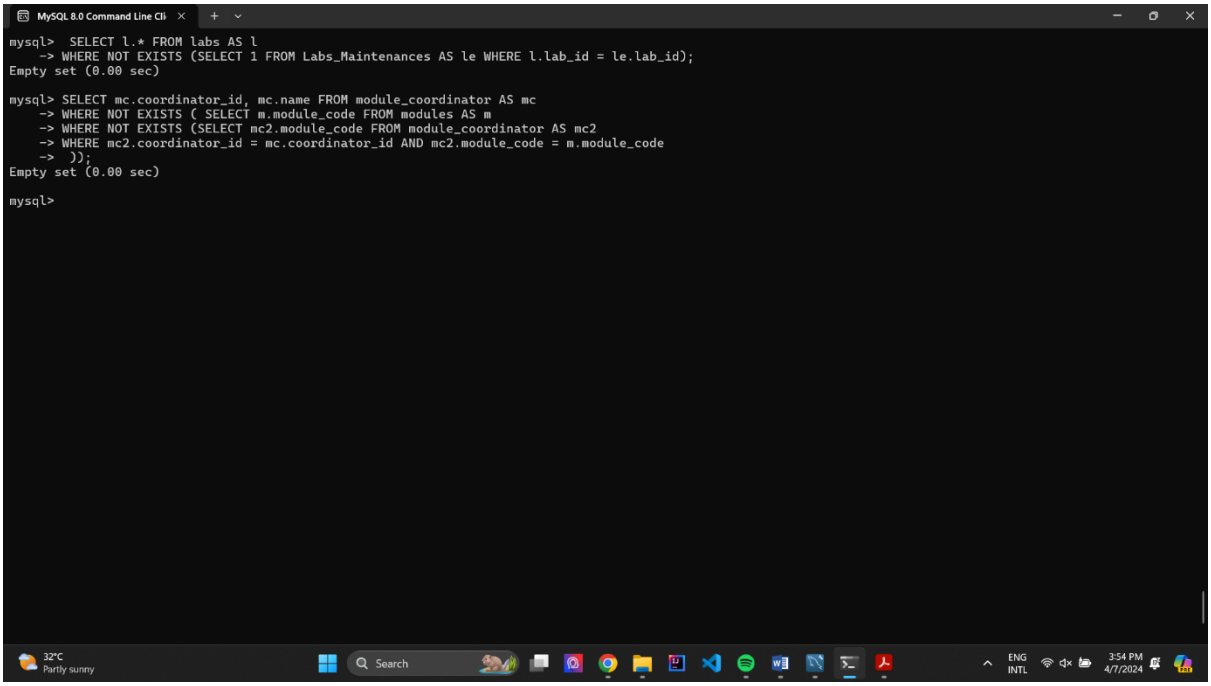


Figure 46 : Division Operation

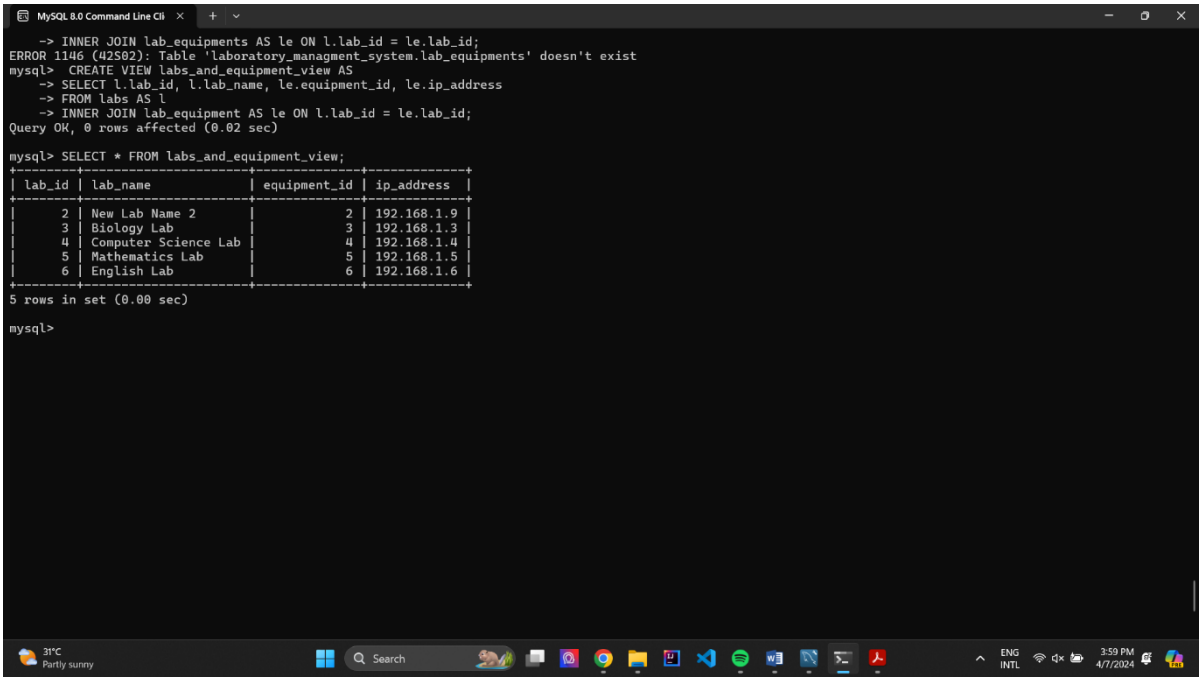


Figure 47 : Inner Join between the Labs and Lab Equipment tables with the user view

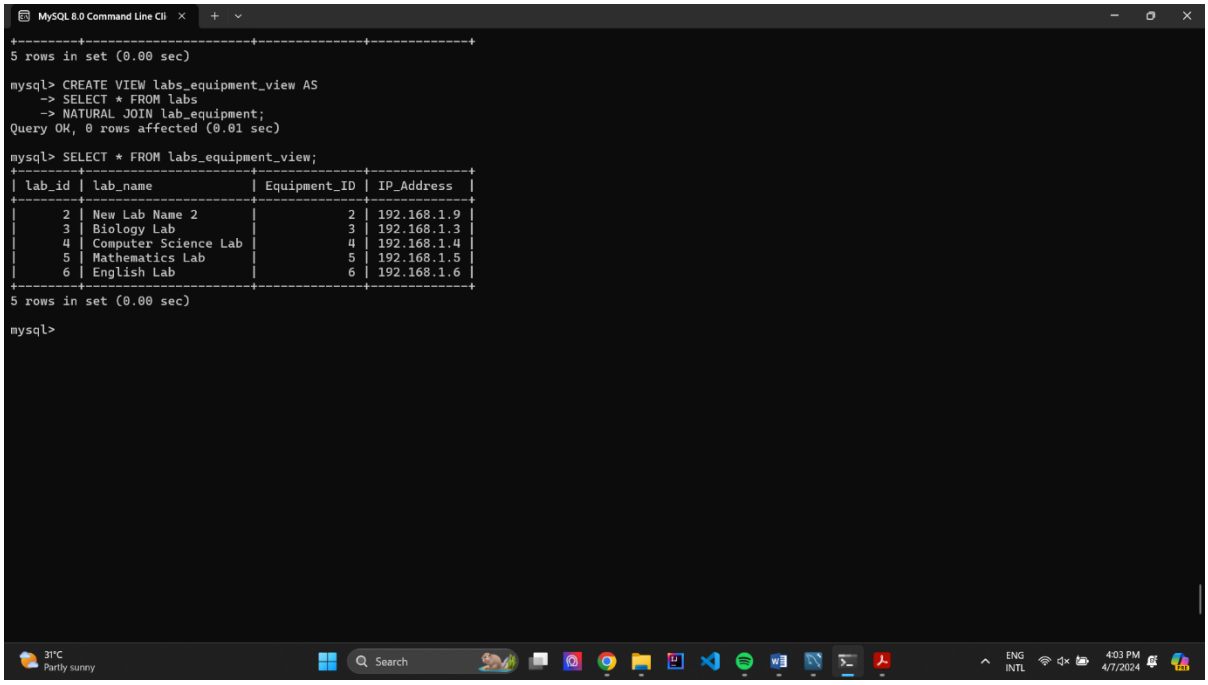


Figure 48 : Natural Join between the Labs and Lab Equipment tables with the user view

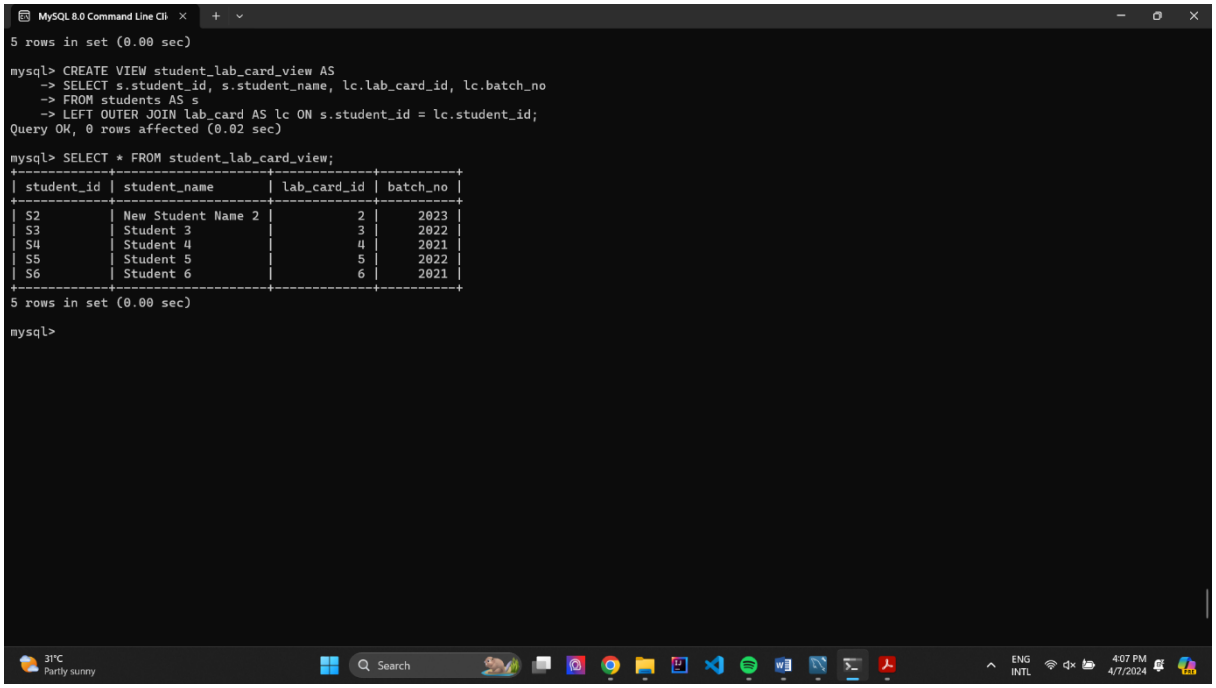


Figure 49 : Left Outer Join between Student and Lab Card tables with user view

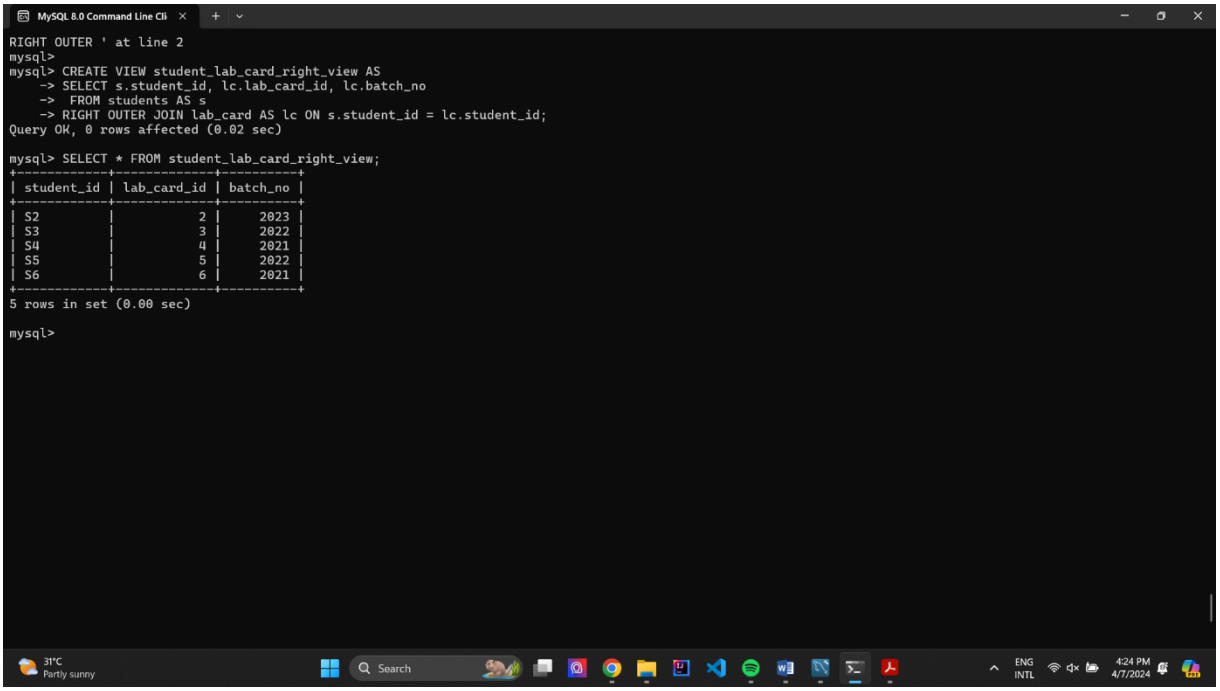


Figure 50 : Right Outer Join between Student and Lab Card tables with user view

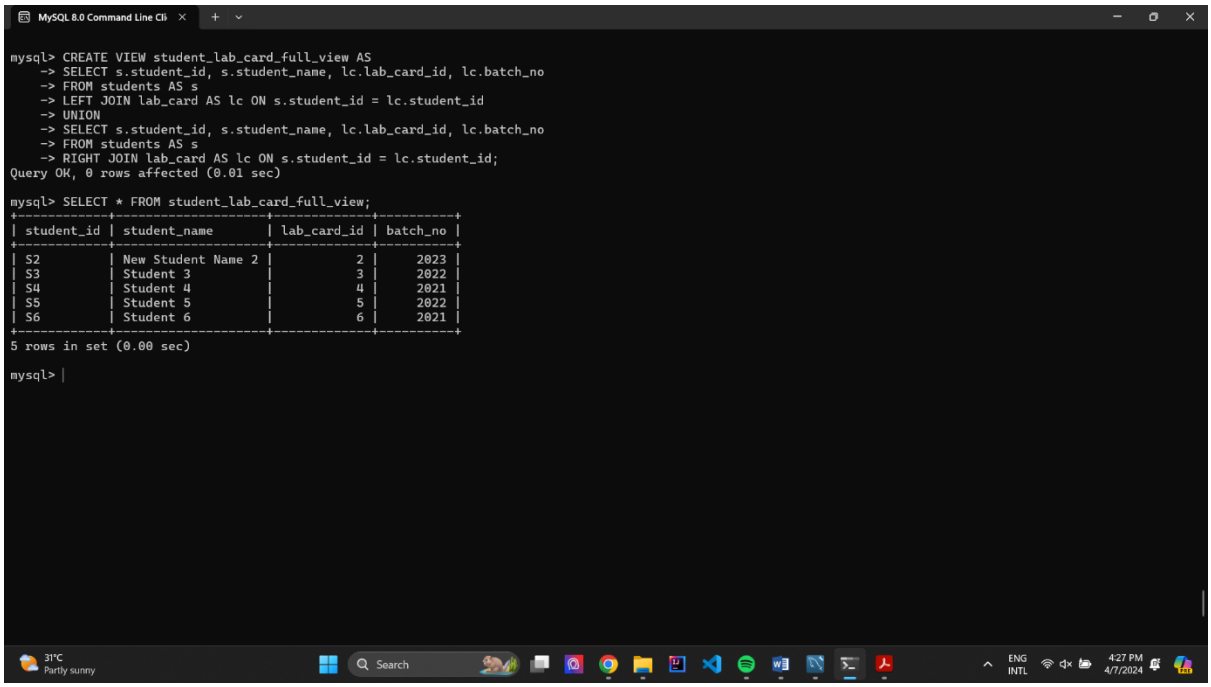


Figure 51 : Full Outer Join between Student and Lab Card tables with user view

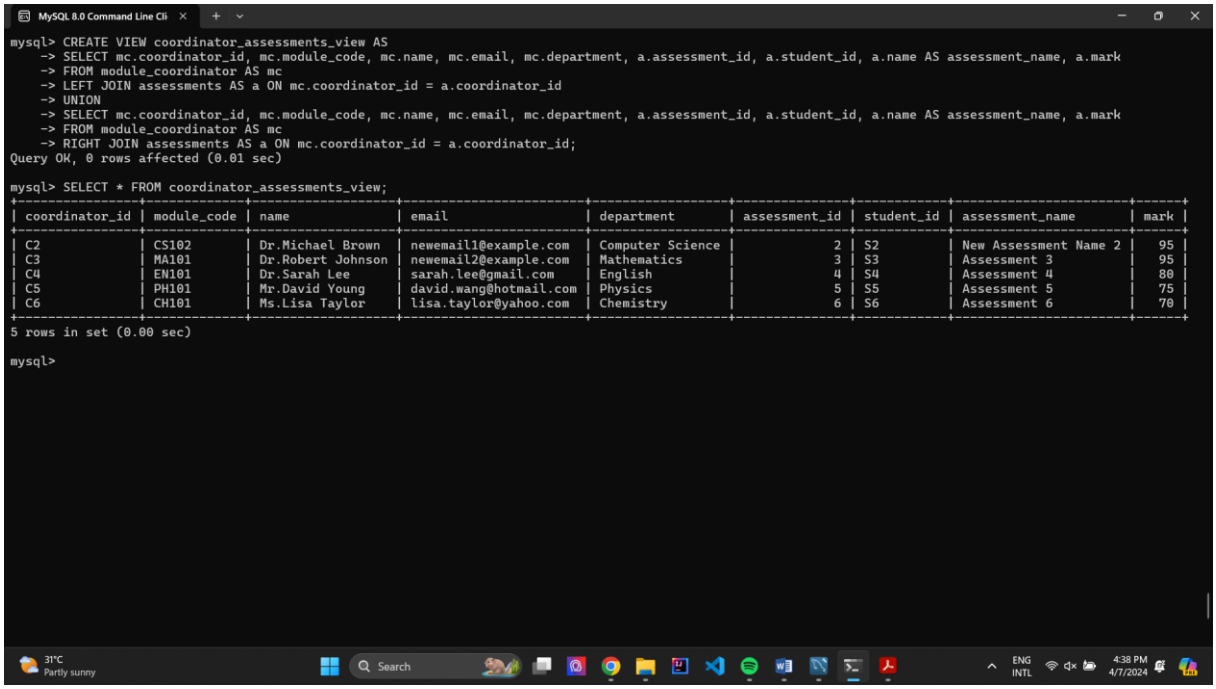


Figure 52 : Full Outer Join between Location and Labs tables with user view

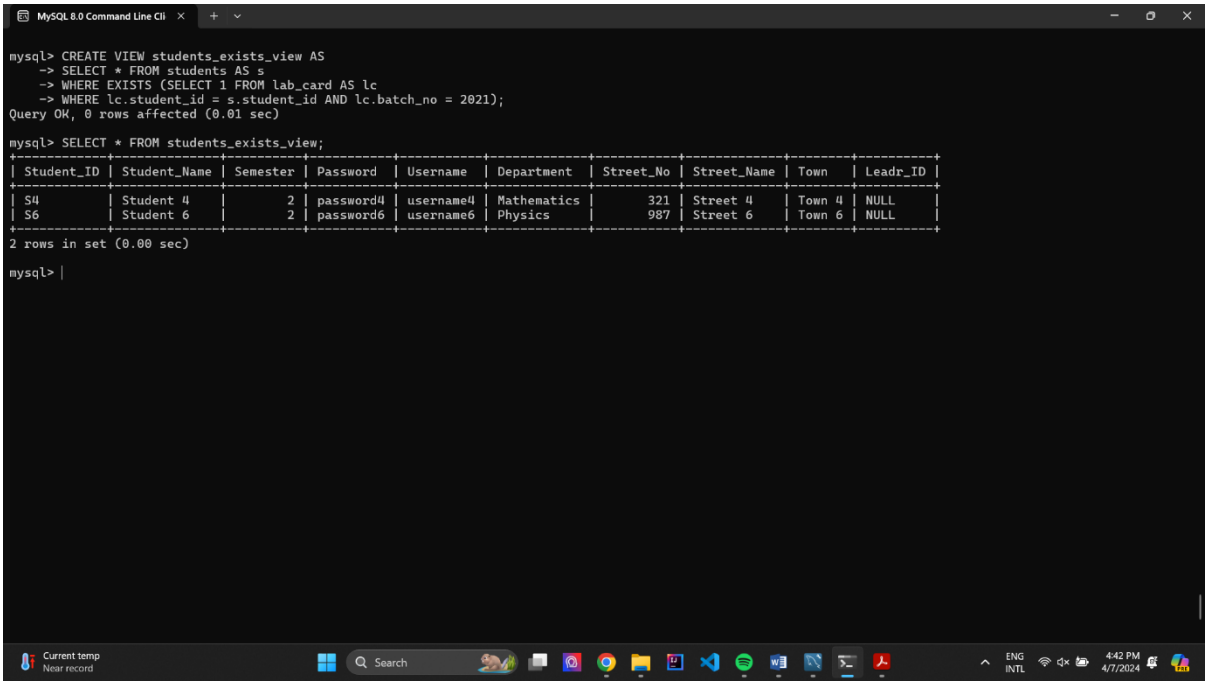


Figure 53 : Outer Union Operation with user view

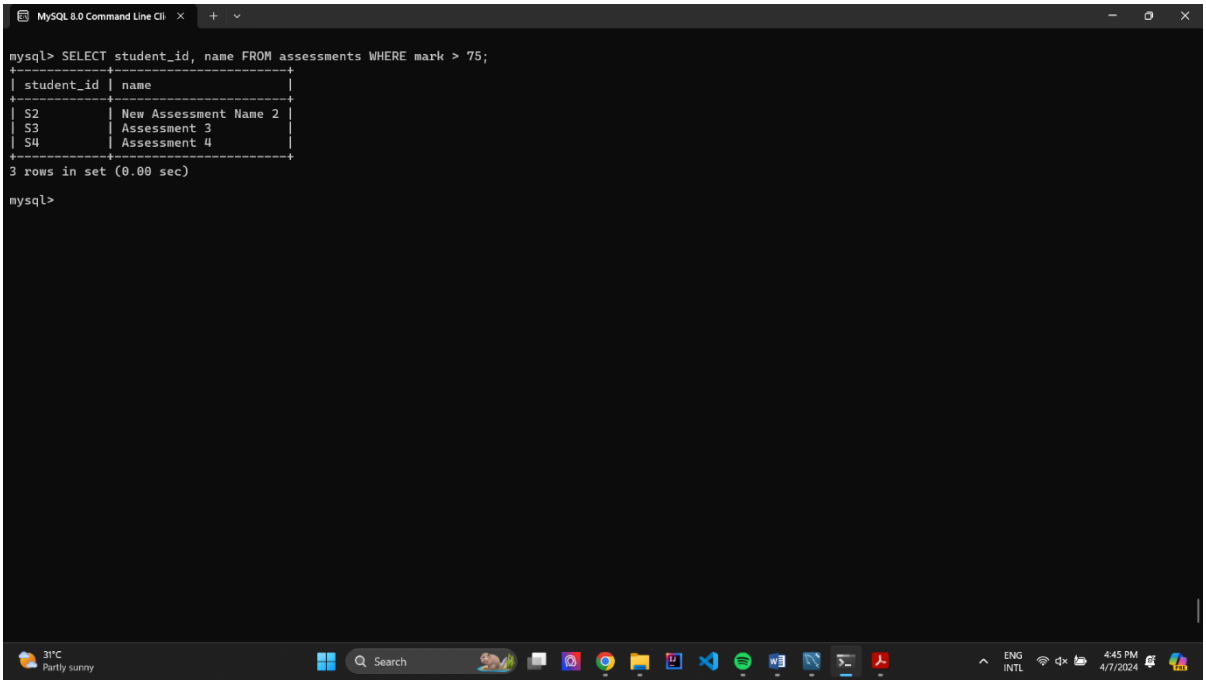


Figure 54 : List of Students Who Scored Above 75

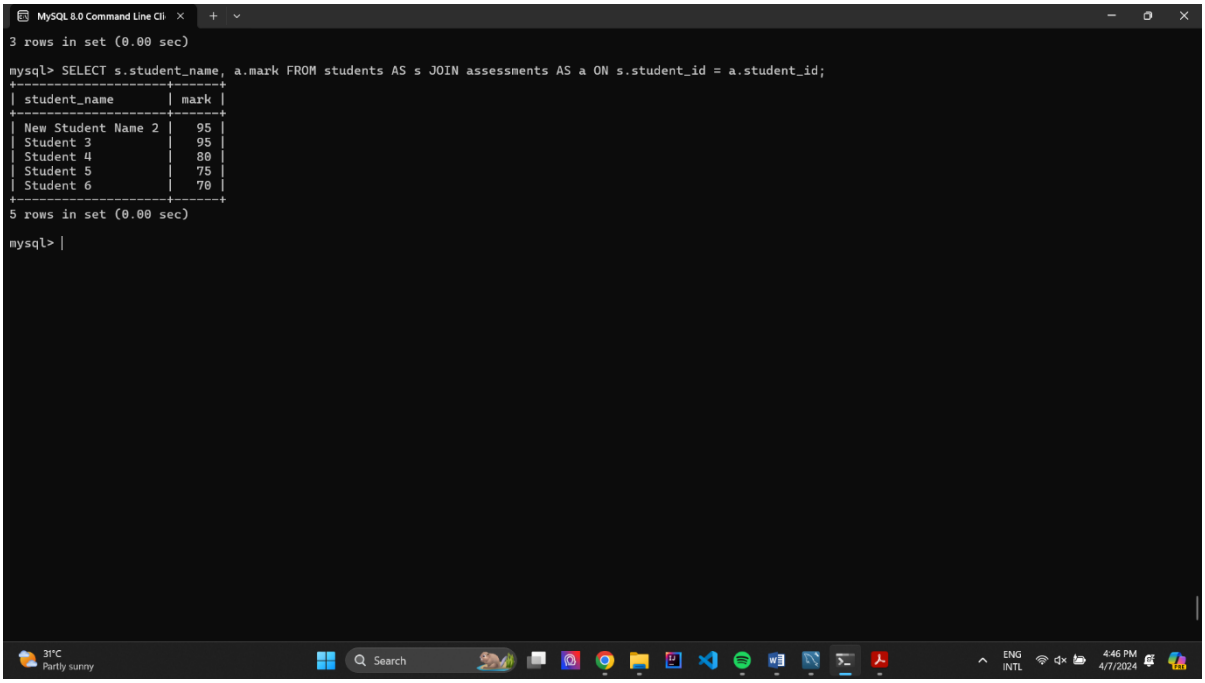


Figure 55 : Student's Assessment Marks

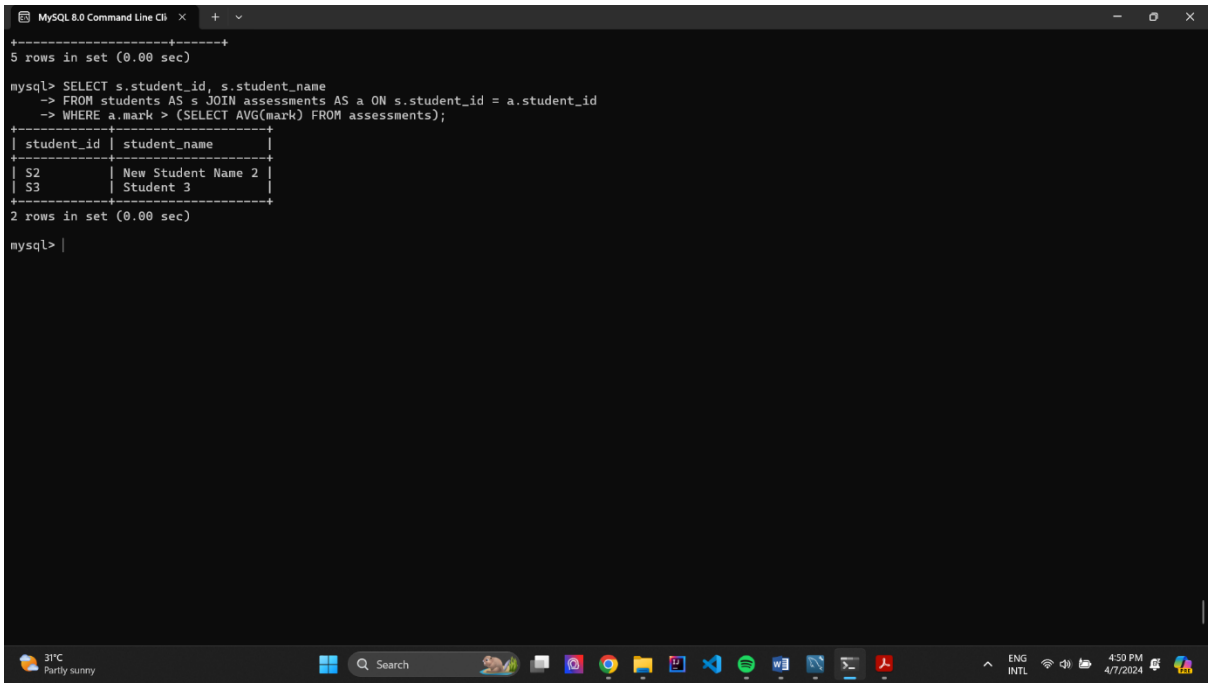


Figure 56 : Students Who Scored Above the Average Mark

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.

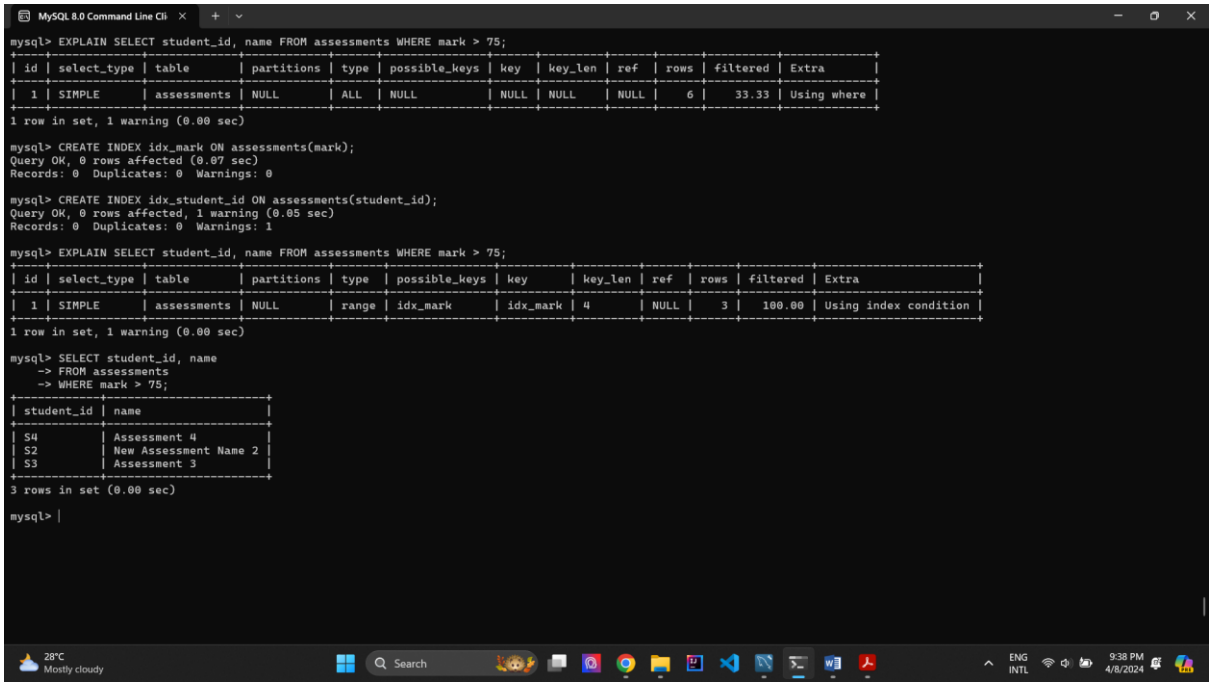


Figure 57 : Tuned Query 1 and Results Comparing

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.

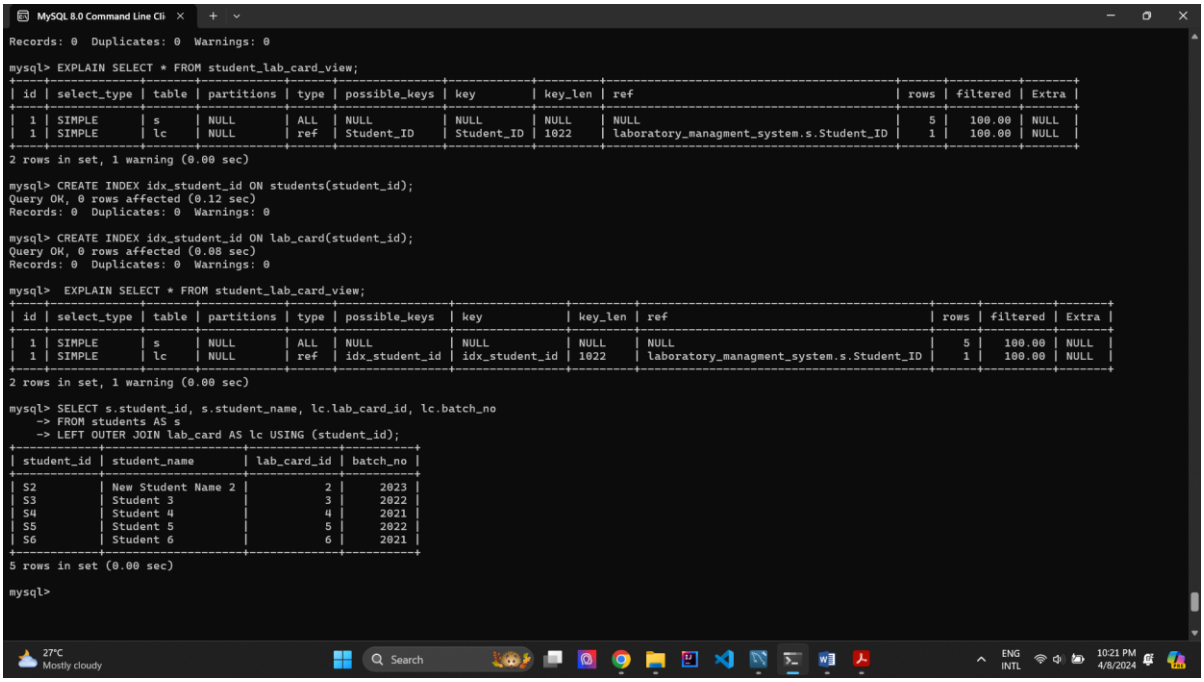


Figure 58 : Tuned Query 2 and Results Comparing

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.

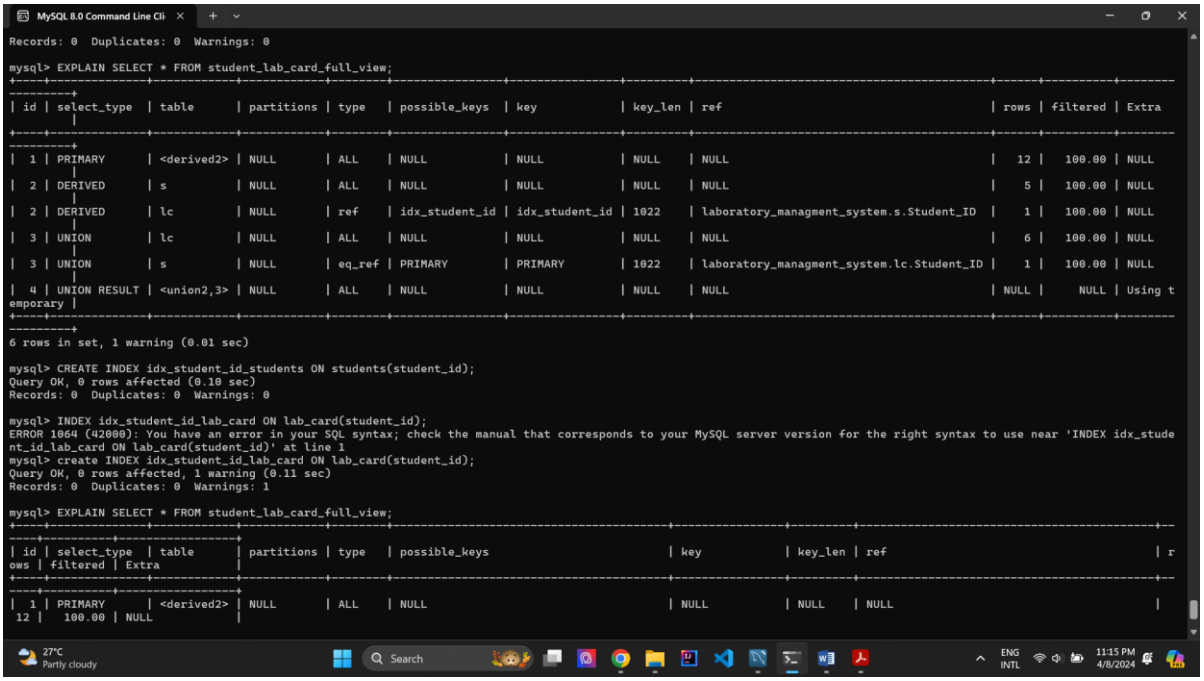


Figure 59 : Results for the Original query

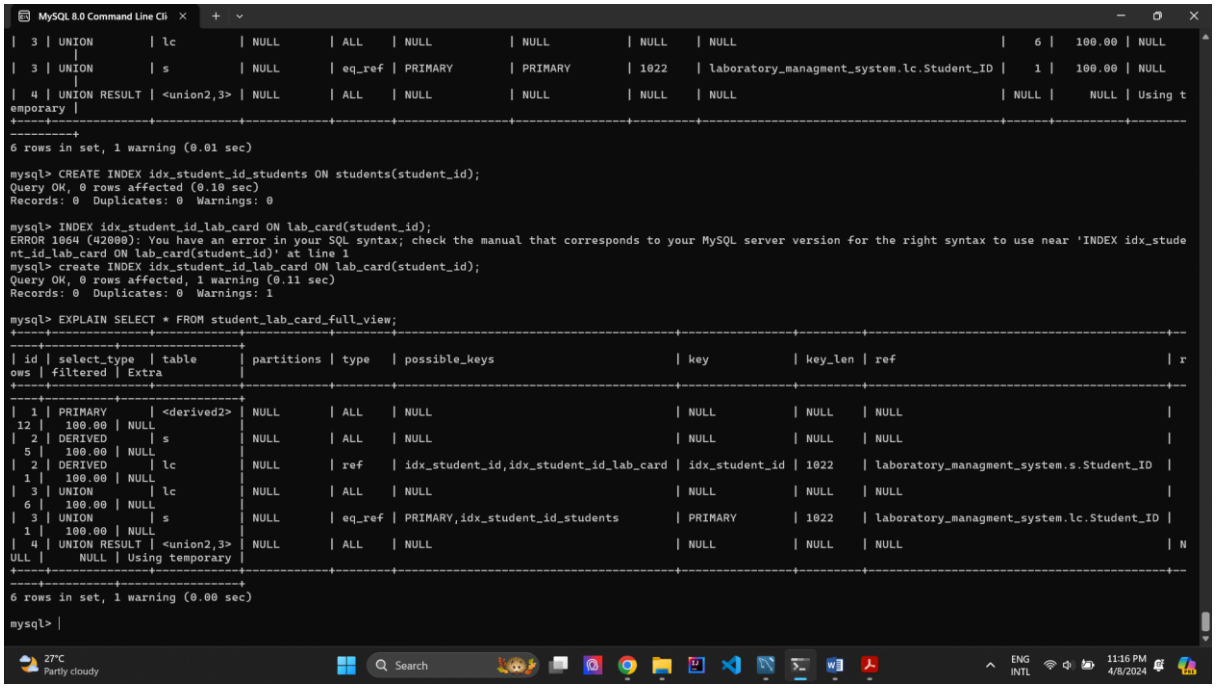


Figure 60 : Results of the Tuned query

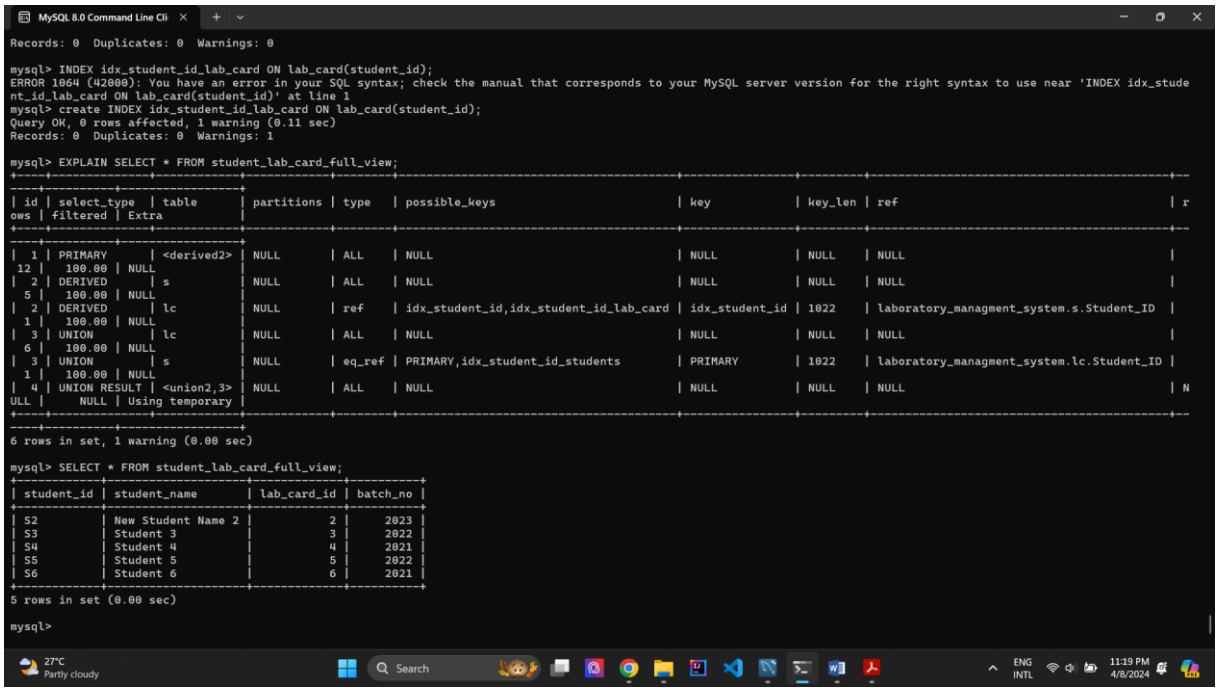


Figure 61 : Tuned Query 3

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.

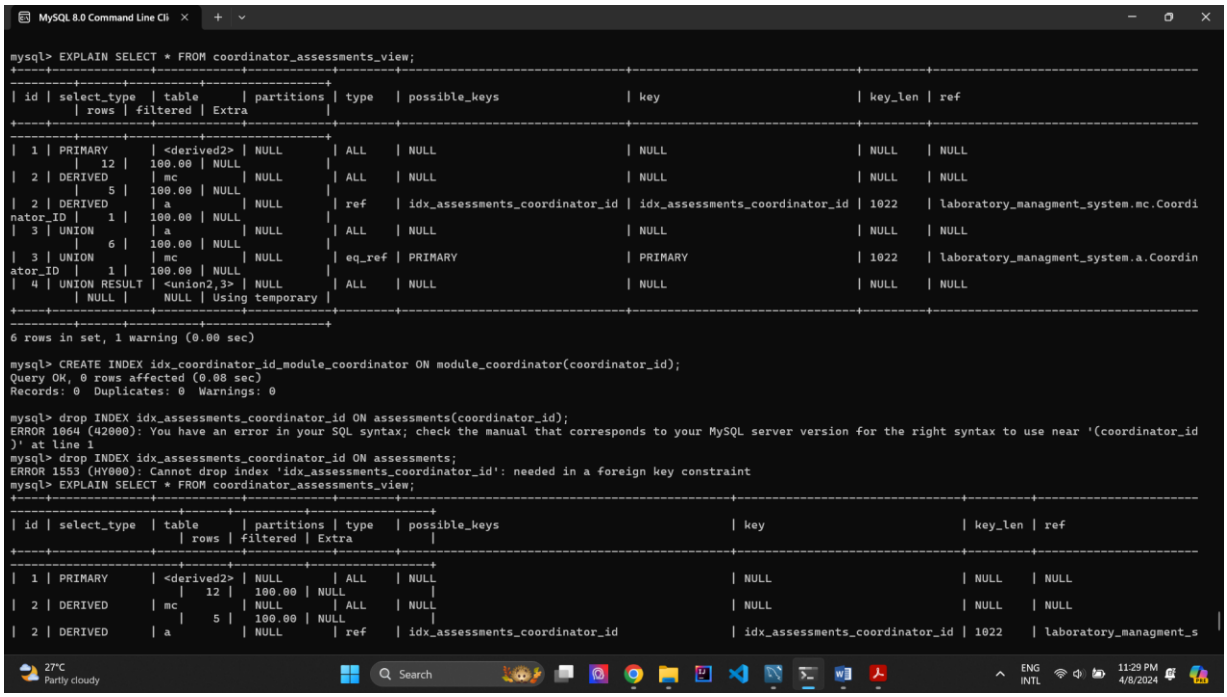


Figure 63 : Results of Original query

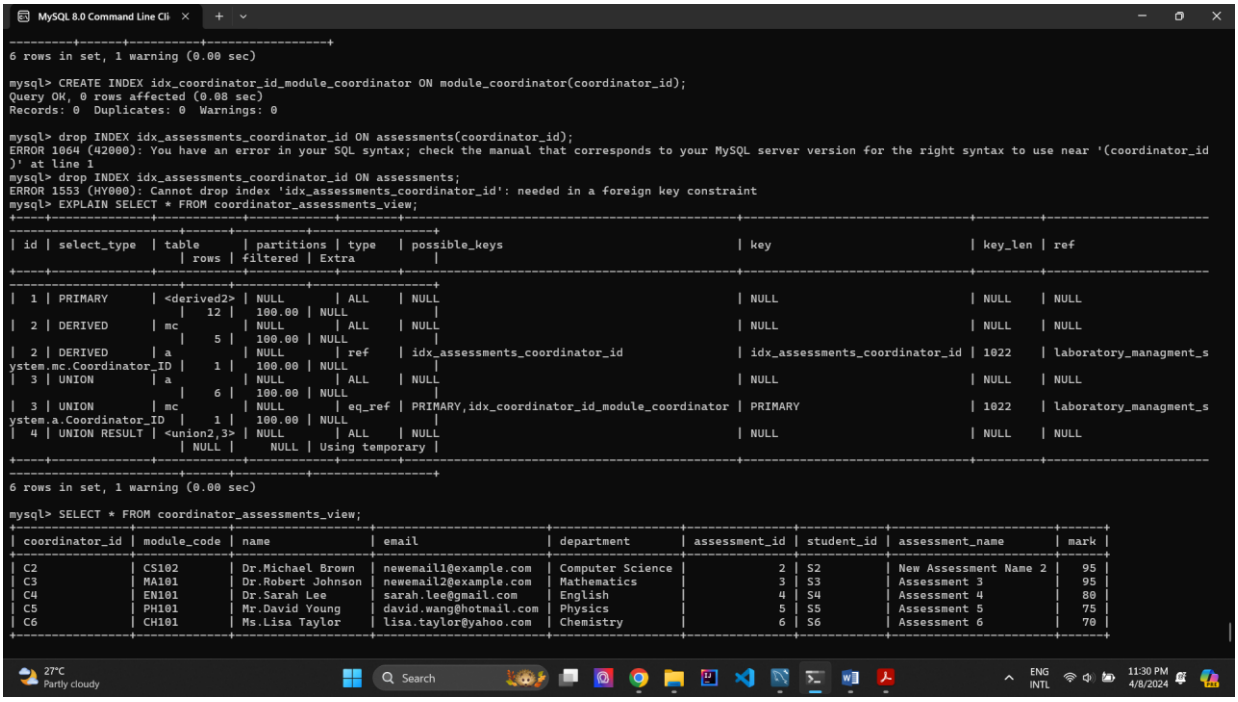


Figure 62 : Tuned query 4 and Results of Tuned query

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.

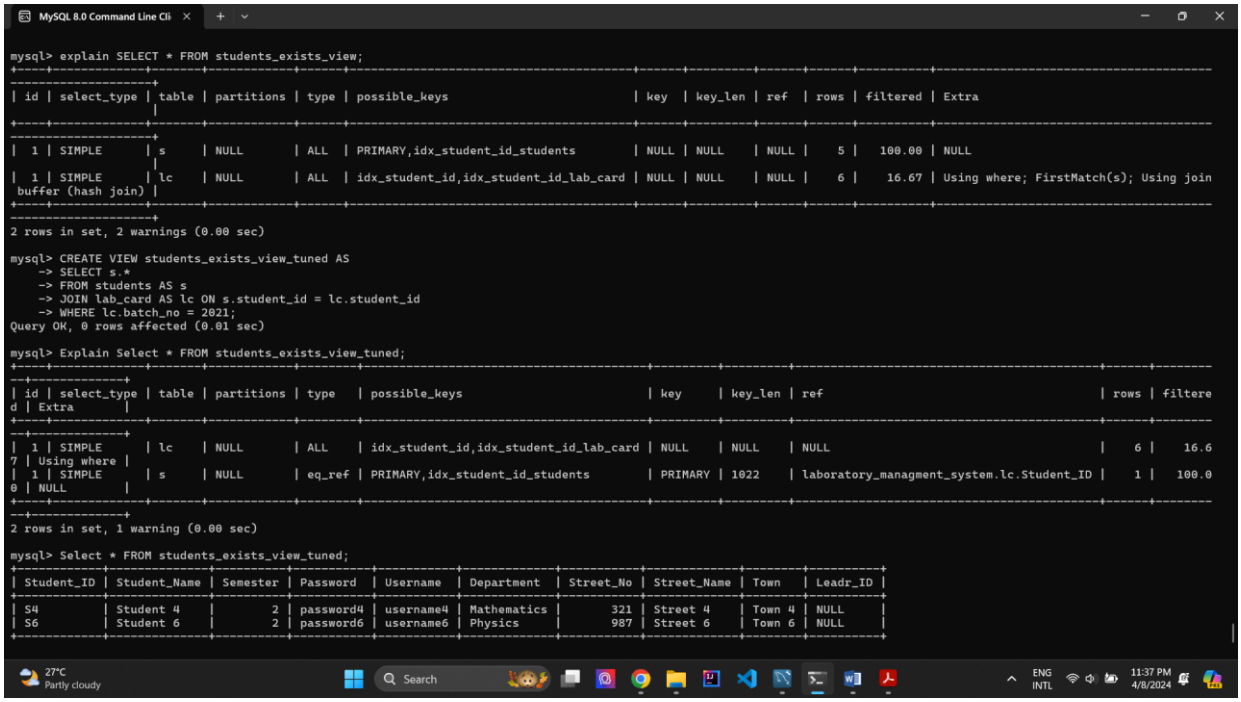


Figure 64 : Tuned query 5 with the Results Comparing

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.

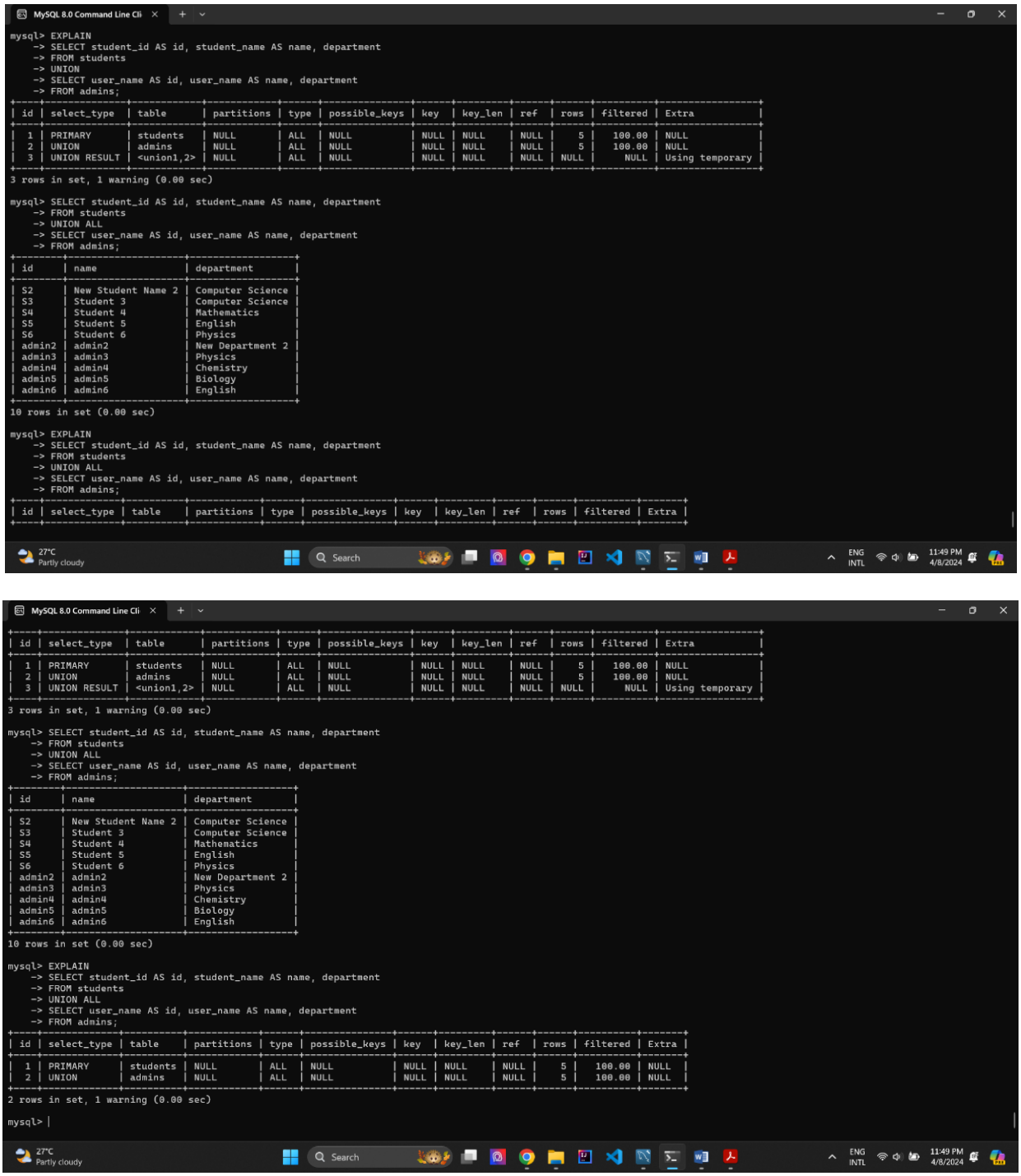


Figure 65 : Tuned query 6 and Results Comparing

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

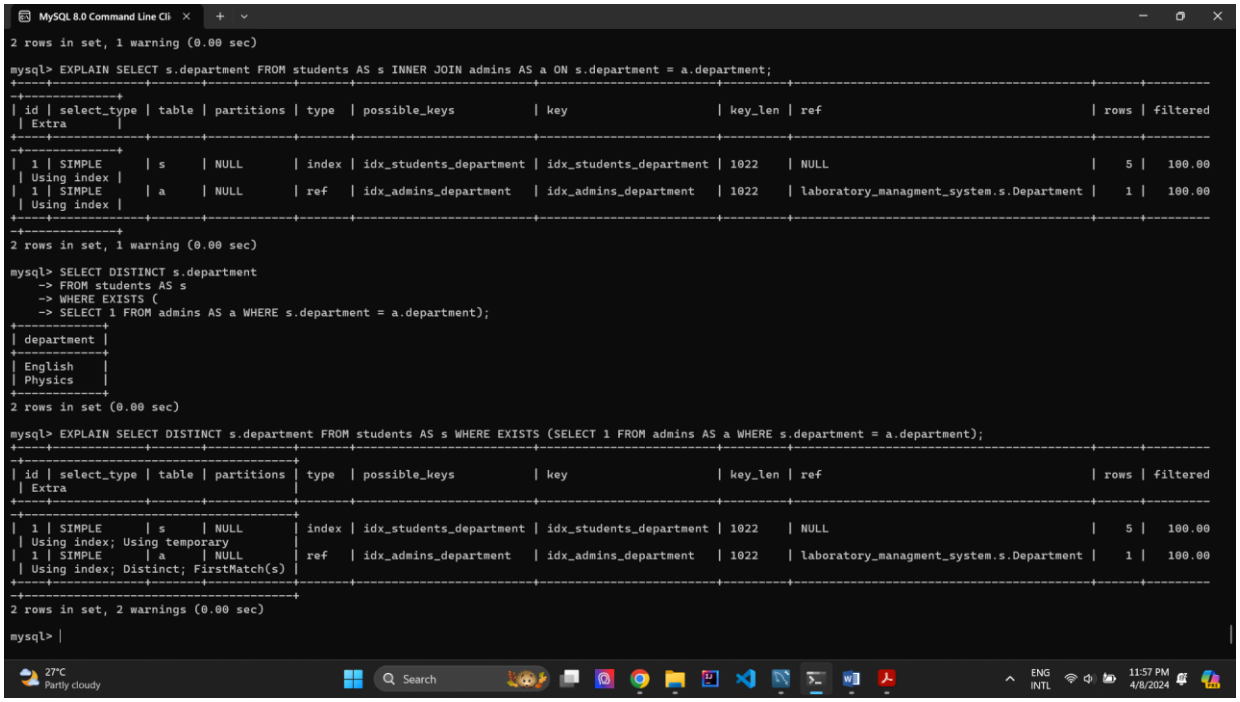


Figure 66 : Tuned query 7 with Results Comparing

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

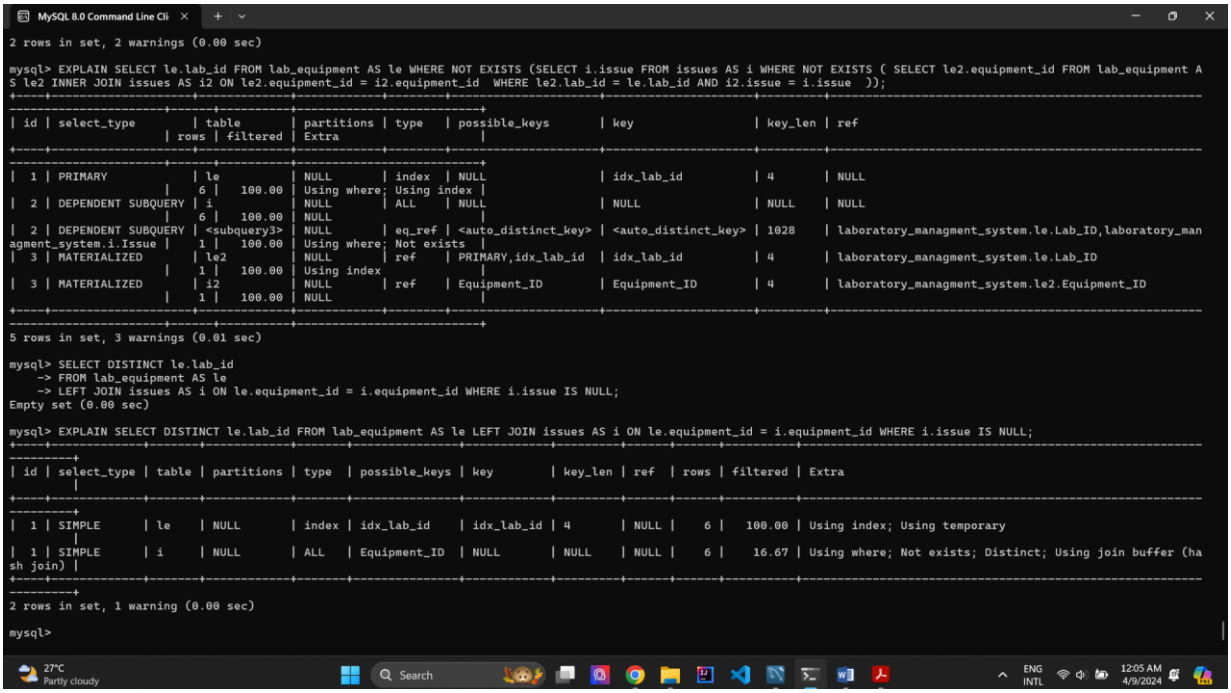
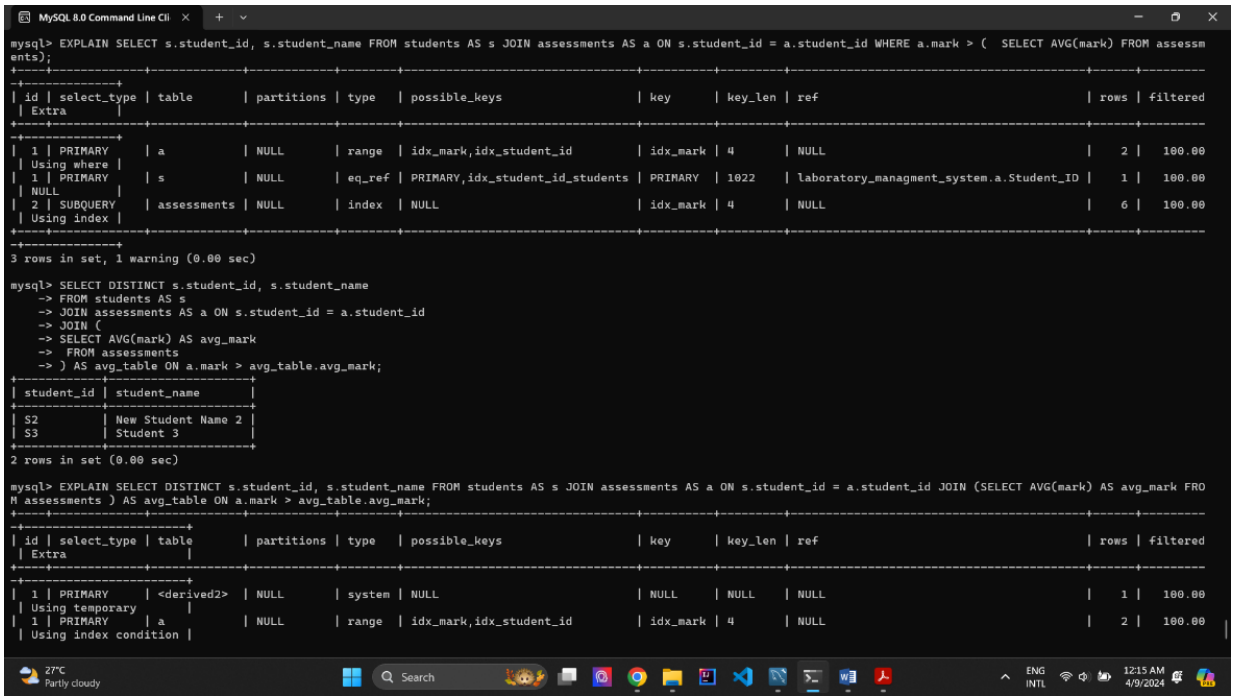


Figure 67 : Tuned query 8 with Results Comparing

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.



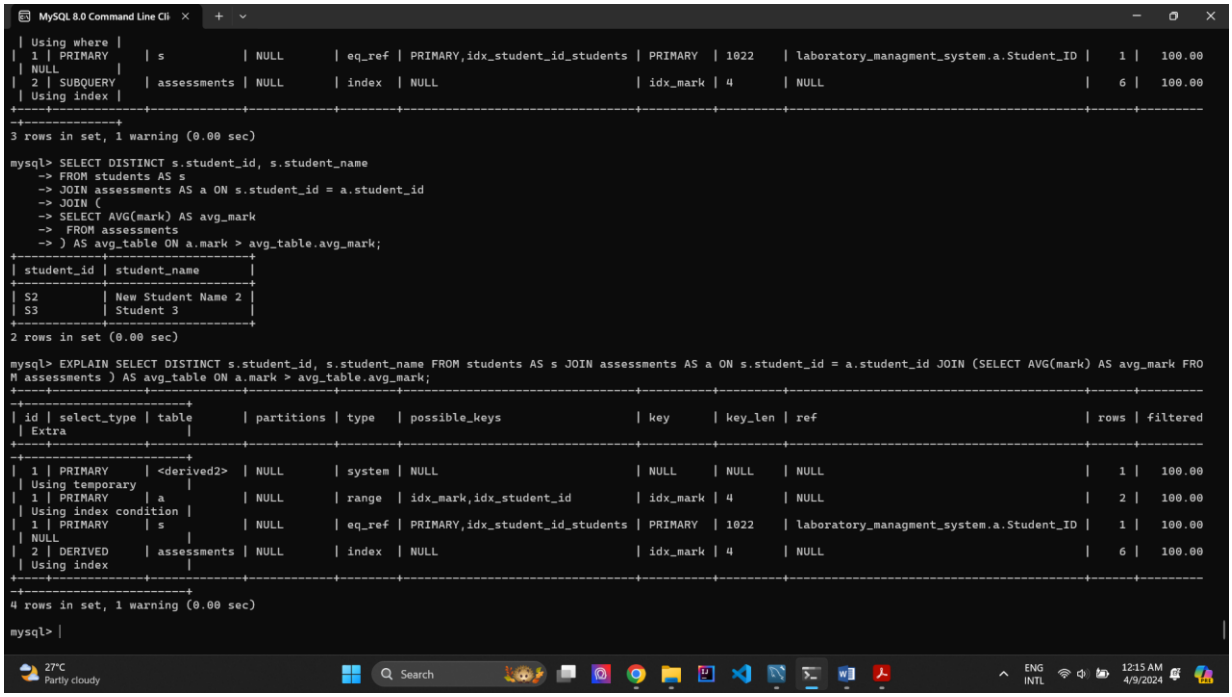


Figure 68 : Tuned query 9 with Results Comparing

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.

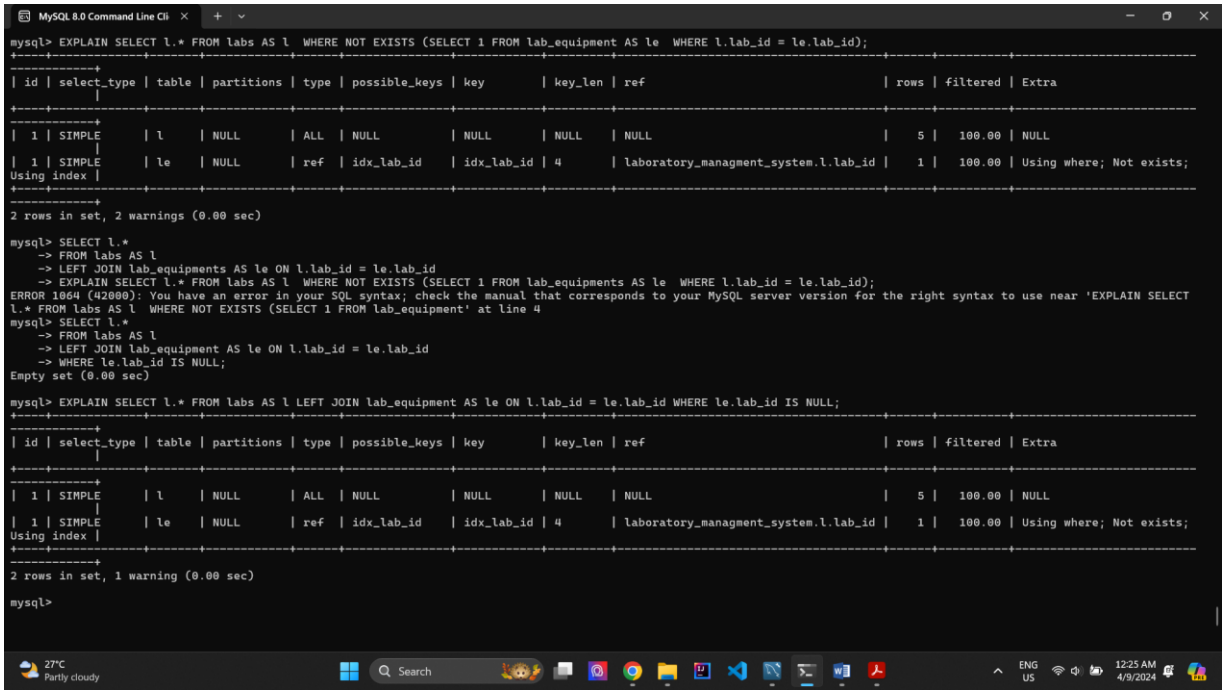


Figure 69 : Tuned query 10 with Results Comparing

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