**Madiha Aimon Tappal**

[**madihaaimon@gmail.com**](mailto:madihaaimon@gmail.com)

**Data Engineering Batch – 1**

**SQL Coding Challenge**

**SQL Join** statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are as follows:

* INNER JOIN
* LEFT JOIN
* RIGHT JOIN
* FULL JOIN
* NATURAL JOIN
  + **INNER JOIN**
* The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.
* **Syntax**:
* SELECT table1.column1, table1.column2, table2.column1....  
  FROM table1   
  INNER JOIN table2  
  ON table1.matching\_column = table2.matching\_column;  
    
    
  **table1**: First table.  
  **table2**: Second table  
  **matching\_column**: Column common to both the tables.



### 1. INNER JOIN:

**Example: Retrieve student information along with their respective courses.**

SELECT

Student.StudentID,

Student.StudentName,

Course.CourseName

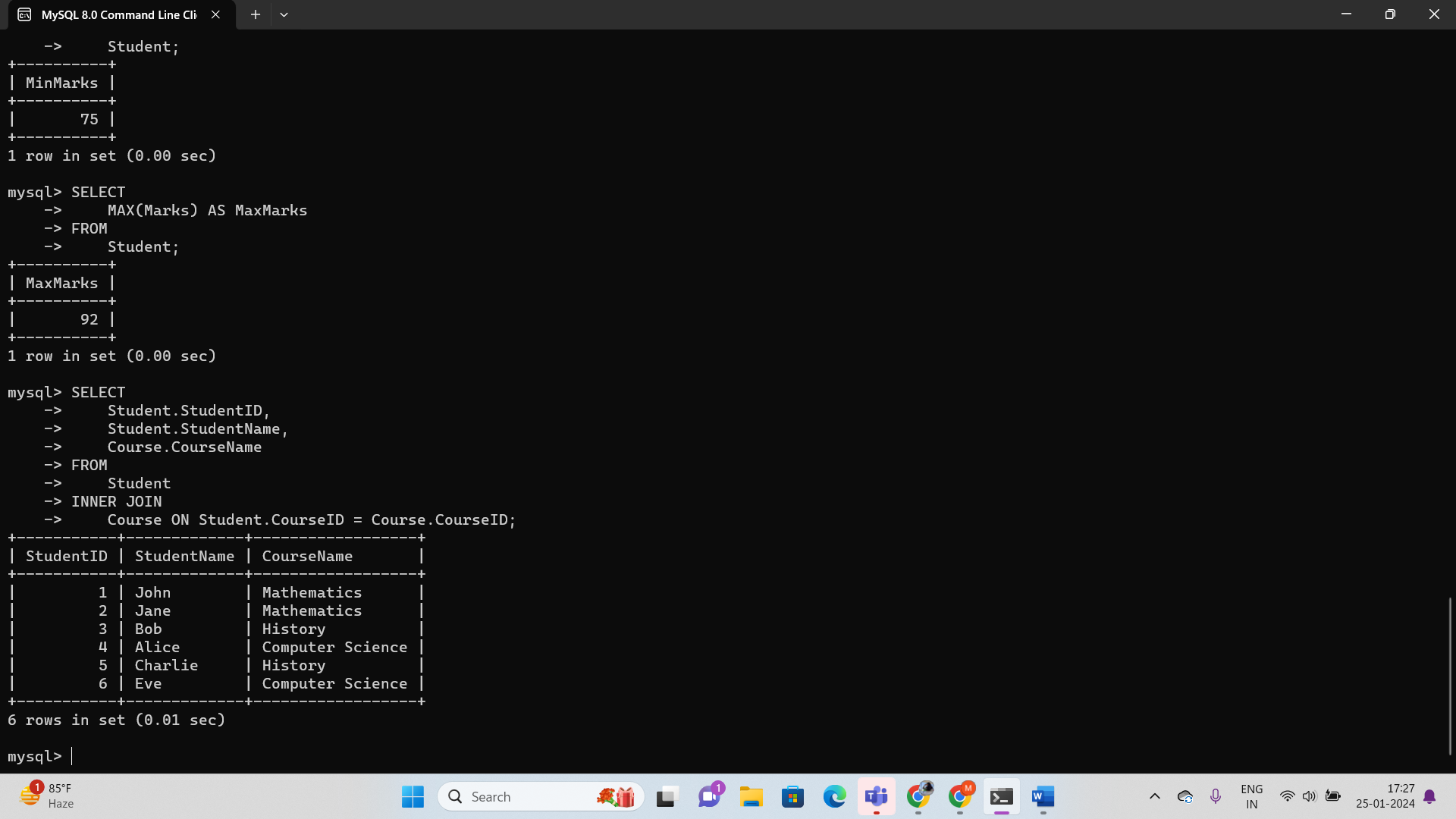
FROM

Student

INNER JOIN

Course ON Student.CourseID = Course.CourseID;

This query uses **INNER JOIN** to combine rows from the **Student** and **Course** tables where there is a match on the **CourseID**. The result includes columns from both tables for matched rows.



### ****LEFT JOIN****

This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain null. LEFT JOIN is also known as LEFT OUTER JOIN.

**Syntax:**

SELECT table1.column1,table1.column2,table2.column1,....  
FROM table1   
LEFT JOIN table2  
ON table1.matching\_column = table2.matching\_column;  
  
  
table1: First table.  
table2: Second table  
matching\_column: Column common to both the tables.



### 2. LEFT JOIN

### ****Example: Retrieve all students and their courses, including those without assigned courses**.**

SELECT

Student.StudentID,

Student.StudentName,

Course.CourseName

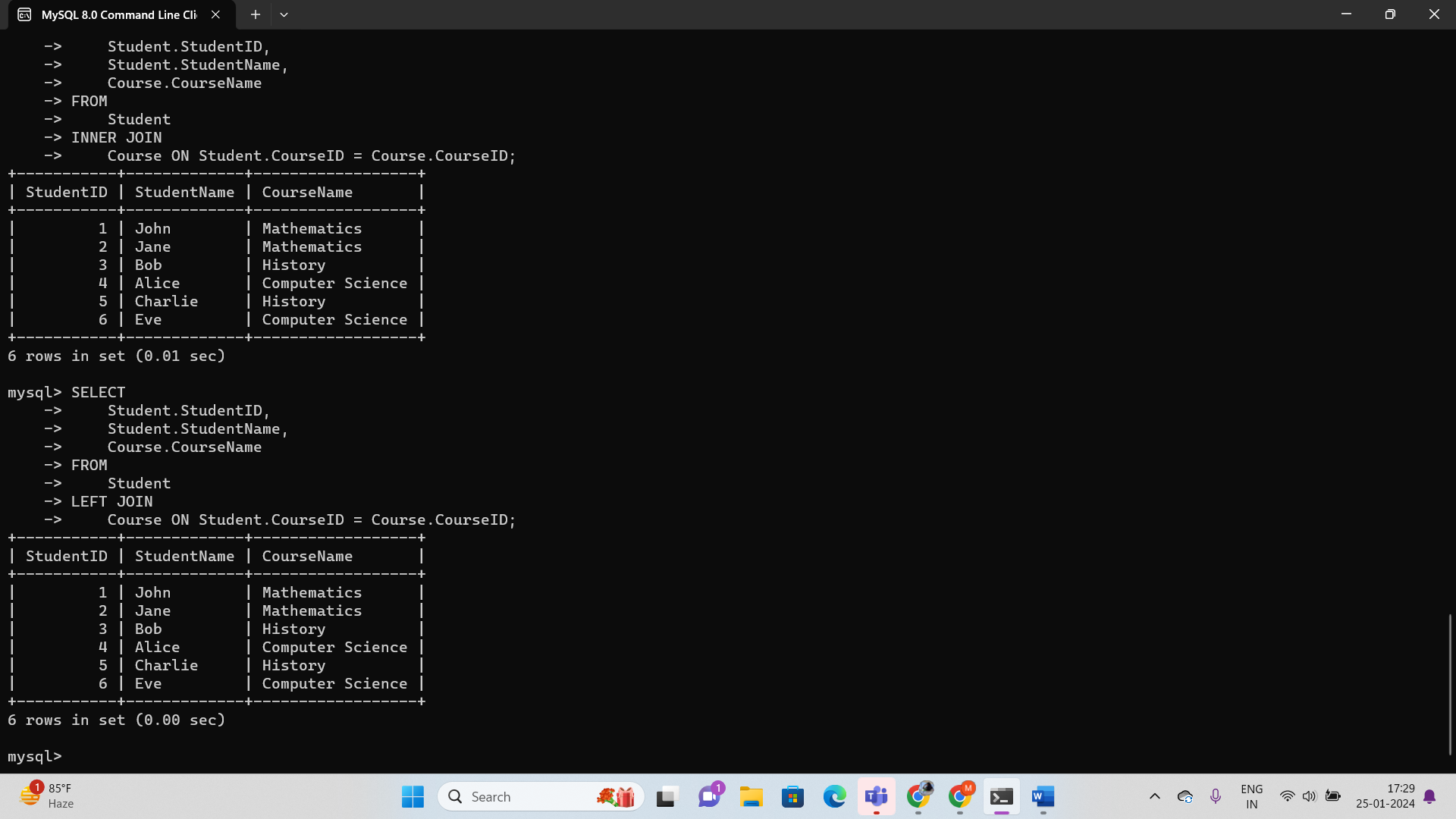
FROM

Student

LEFT JOIN

Course ON Student.CourseID = Course.CourseID;

This query uses **LEFT JOIN** to retrieve all rows from the **Student** table and the matching rows from the **Course** table. If a student doesn't have an assigned course, the columns from the **Course** table will contain **NULL**.

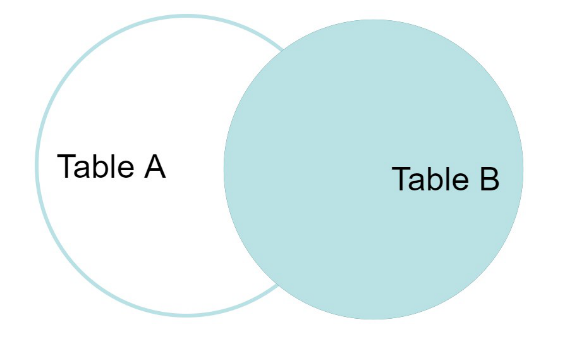


### ****RIGHT JOIN****

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain null. RIGHT JOIN is also known as RIGHT OUTER JOIN.

**Syntax:**

SELECT table1.column1, table1.column2, table2.column1,.  
FROM table1   
RIGHT JOIN table2  
ON table1.matching\_column = table2.matching\_column;  
  
  
table1: First table.  
table2: Second table  
matching\_column: Column common to both the tables.



### 3. RIGHT JOIN

**Example: Retrieve all courses and the students enrolled in each course, including courses without any students.**

SELECT

Student.StudentID,

Student.StudentName,

Course.CourseName

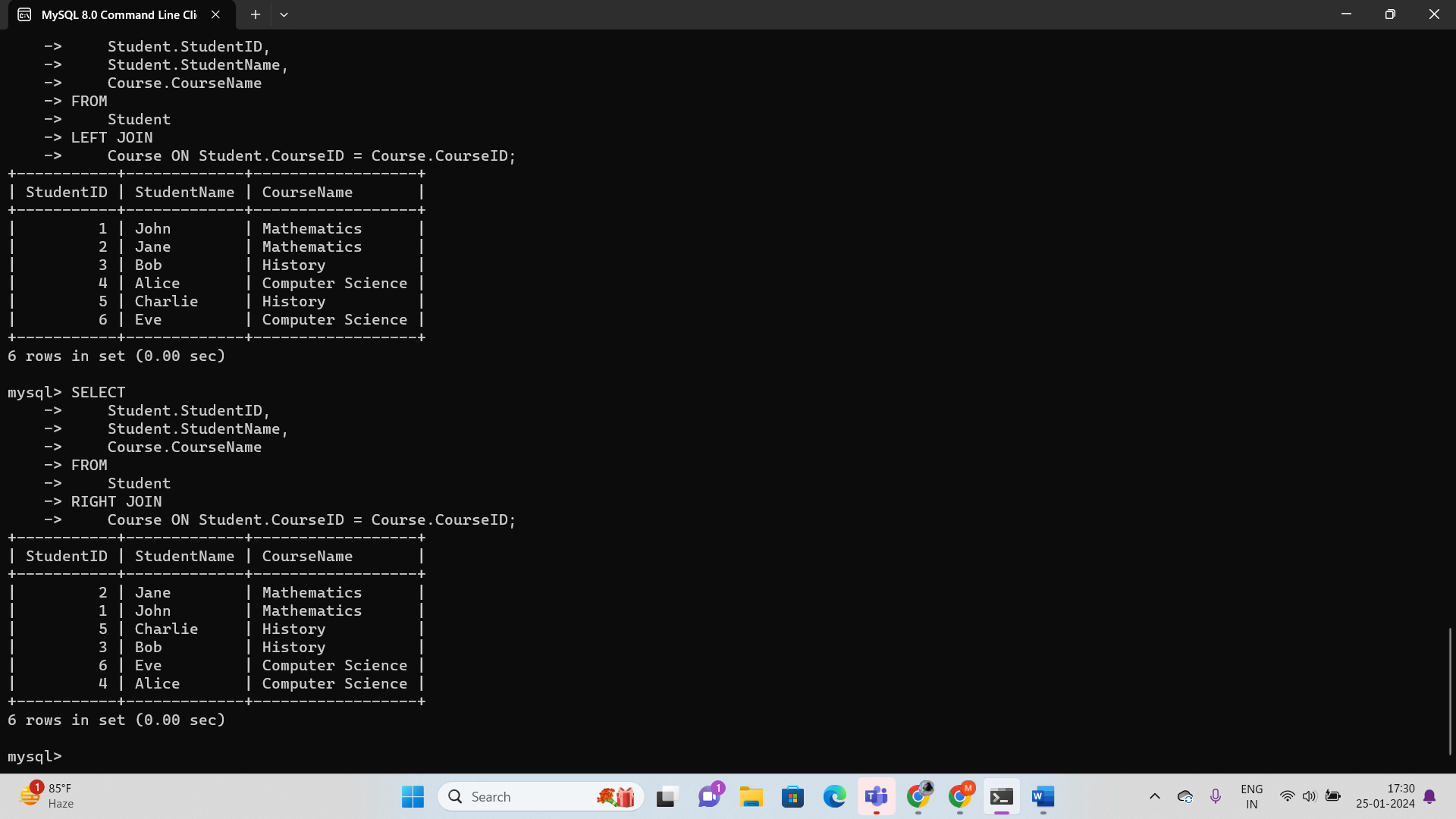
FROM

Student

RIGHT JOIN

Course ON Student.CourseID = Course.CourseID;

This query uses **RIGHT JOIN** to retrieve all rows from the **Course** table and the matching rows from the **Student** table. If a course has no enrolled students, the columns from the **Student** table will contain **NULL**.



### ****FULL JOIN****

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain NULL values.

**Syntax:**

SELECT table1.column1, table1.column2, table2.column1,.  
FROM table1   
FULL JOIN table2  
ON table1.matching\_column = table2.matching\_column;  
  
  
table1: First table.  
table2: Second table  
matching\_column: Column common to both the tables.



### 4. FULL JOIN

**Example: Retrieve all students and their courses, including unmatched students and courses.**

SELECT

Student.StudentID,

Student.StudentName,

Course.CourseName

FROM

Student

FULL JOIN

Course ON Student.CourseID = Course.CourseID;

In MySQL, there is no direct support for **FULL JOIN**. Instead, you can achieve the same result using a combination of **LEFT JOIN** and **UNION** with a **RIGHT JOIN**.

-- Records from Student that have corresponding courses

SELECT

Student.StudentID,

Student.StudentName,

Course.CourseName

FROM

Student

LEFT JOIN

Course ON Student.CourseID = Course.CourseID

UNION

-- Records from Course that don't have corresponding students

SELECT

Student.StudentID,

Student.StudentName,

Course.CourseName

FROM

Student

RIGHT JOIN

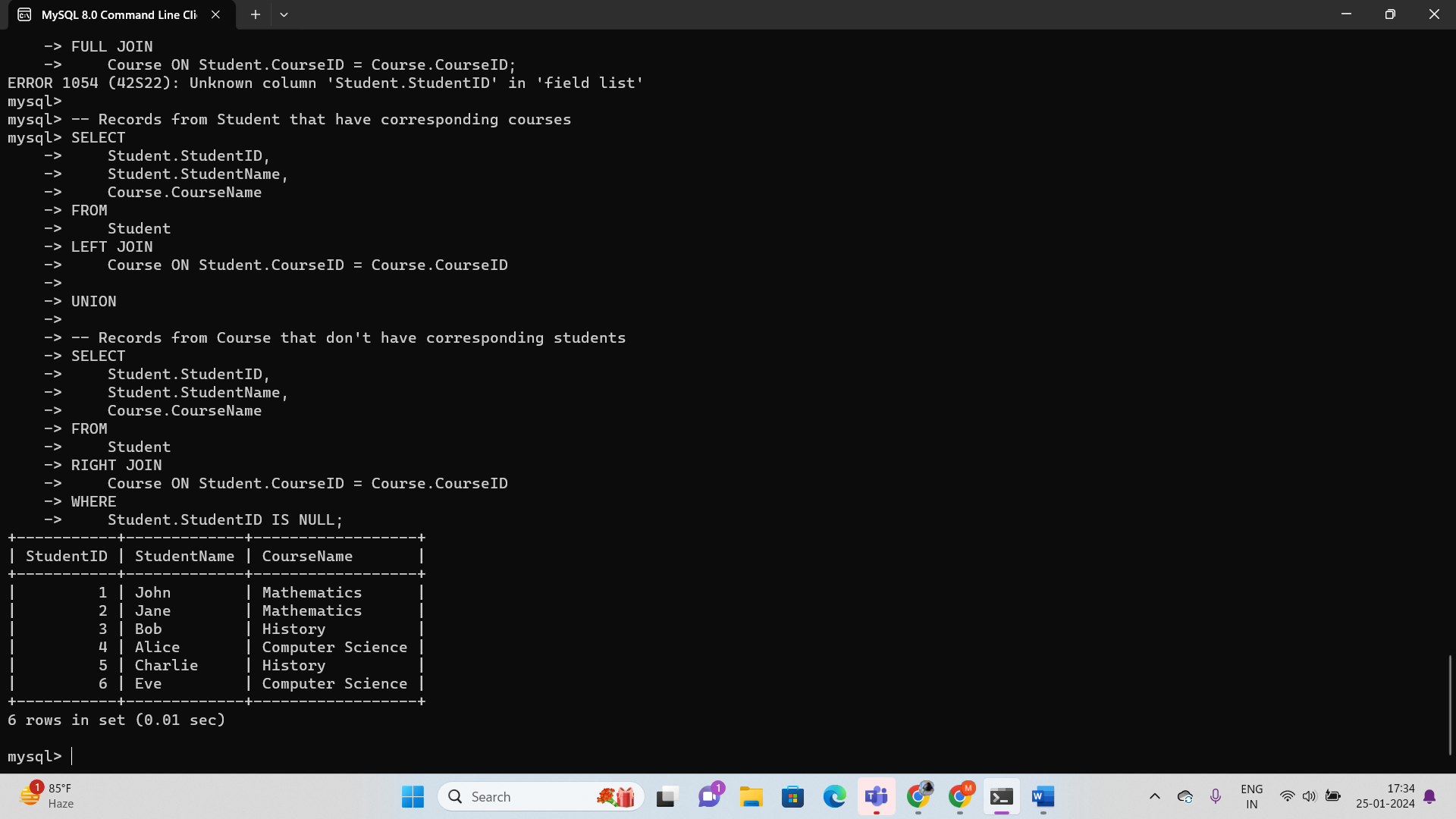
Course ON Student.CourseID = Course.CourseID

WHERE

Student.StudentID IS NULL;

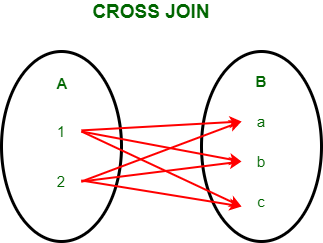
In this query:

* The **LEFT JOIN** retrieves records from **Student** that have corresponding courses.
* The **RIGHT JOIN** retrieves records from **Course** that don't have corresponding students.
* The **UNION** combines the results of the two queries.



### CROSS join

In SQL, a Cross Join is also called a Cartesian Join, it performs cross product of records of two or more joined tables. Sometimes we need to match each row of one table to every other row of another table so in this case **cross Join** is the best choice. Performing a cross is helpful in many applications where we need to obtain paired combinations of records.



*Cross Join*

**Syntax of CROSS JOIN:**

Cross-join allows us to join each and every row of both tables. It is similar to the Cartesian product that joins all the rows. Below is the syntax of cross-join.

*SELECT \**

*FROM table1*

*CROSS JOIN table2;*

**5. CROSS JOIN:**

**Example: Generate all possible combinations of students and courses.**

SELECT

Student.StudentID,

Student.StudentName,

Course.CourseID,

Course.CourseName

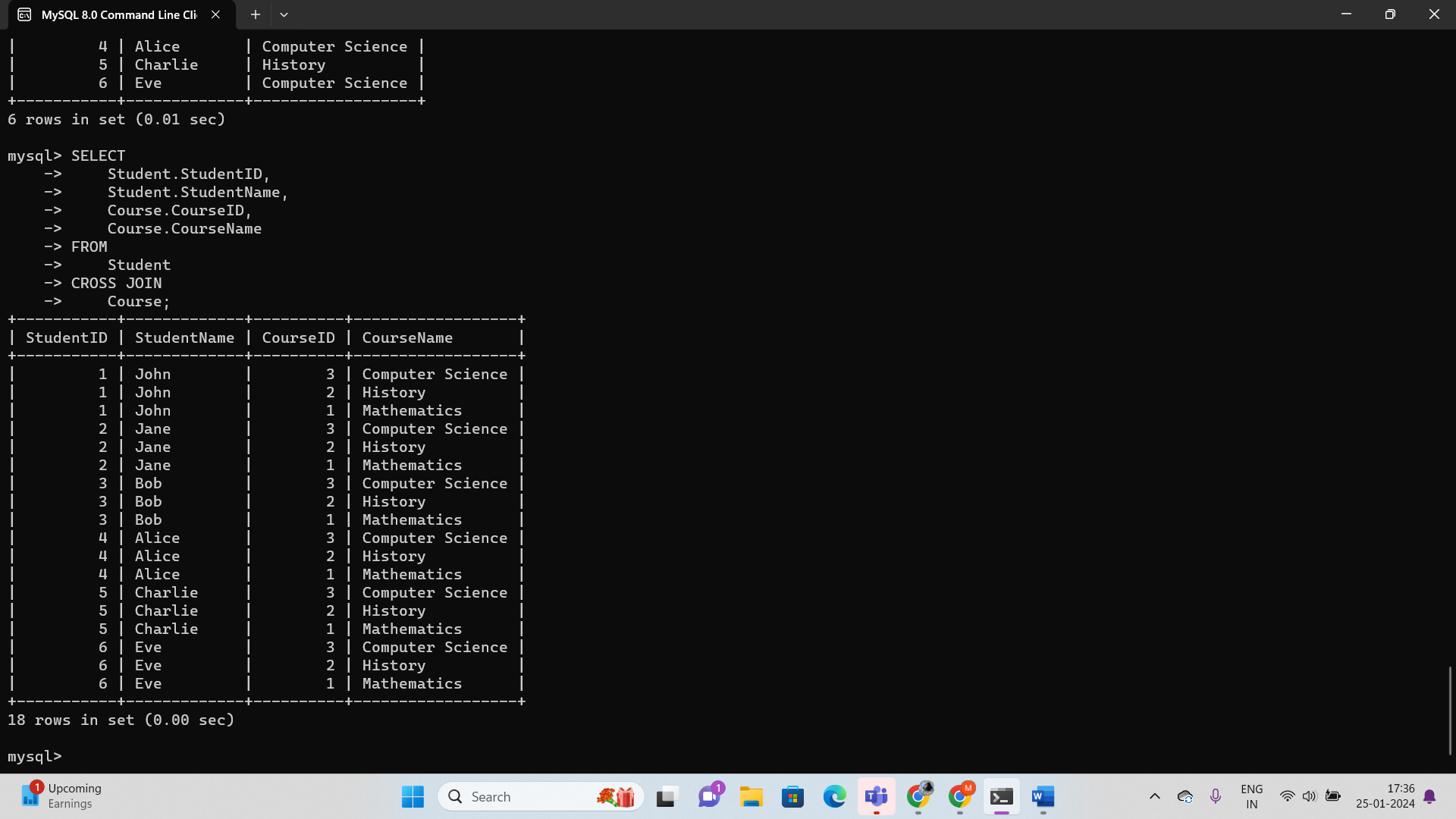
FROM

Student

CROSS JOIN

Course;

This query uses **CROSS JOIN** to create the Cartesian product of the **Student** and **Course** tables, resulting in all possible combinations of students and courses.



### Natural join

Natural join can join tables based on the common columns in the tables being joined. A natural join returns all rows by matching values in common columns having same name and data type of columns and that column should be present in both tables.

Both tables must have at list one common column with same column name and same data type.

**6. Natural join**

A **NATURAL JOIN** is a type of SQL join that automatically matches columns with the same name in both tables, without the need for explicitly specifying the columns to join on. It is essentially a shorthand for writing an **INNER JOIN** with a specific condition.

Here's an example of using **NATURAL JOIN** with the **Student** and **Course** tables:

SELECT

Student.StudentID,

Student.StudentName,

Course.CourseID,

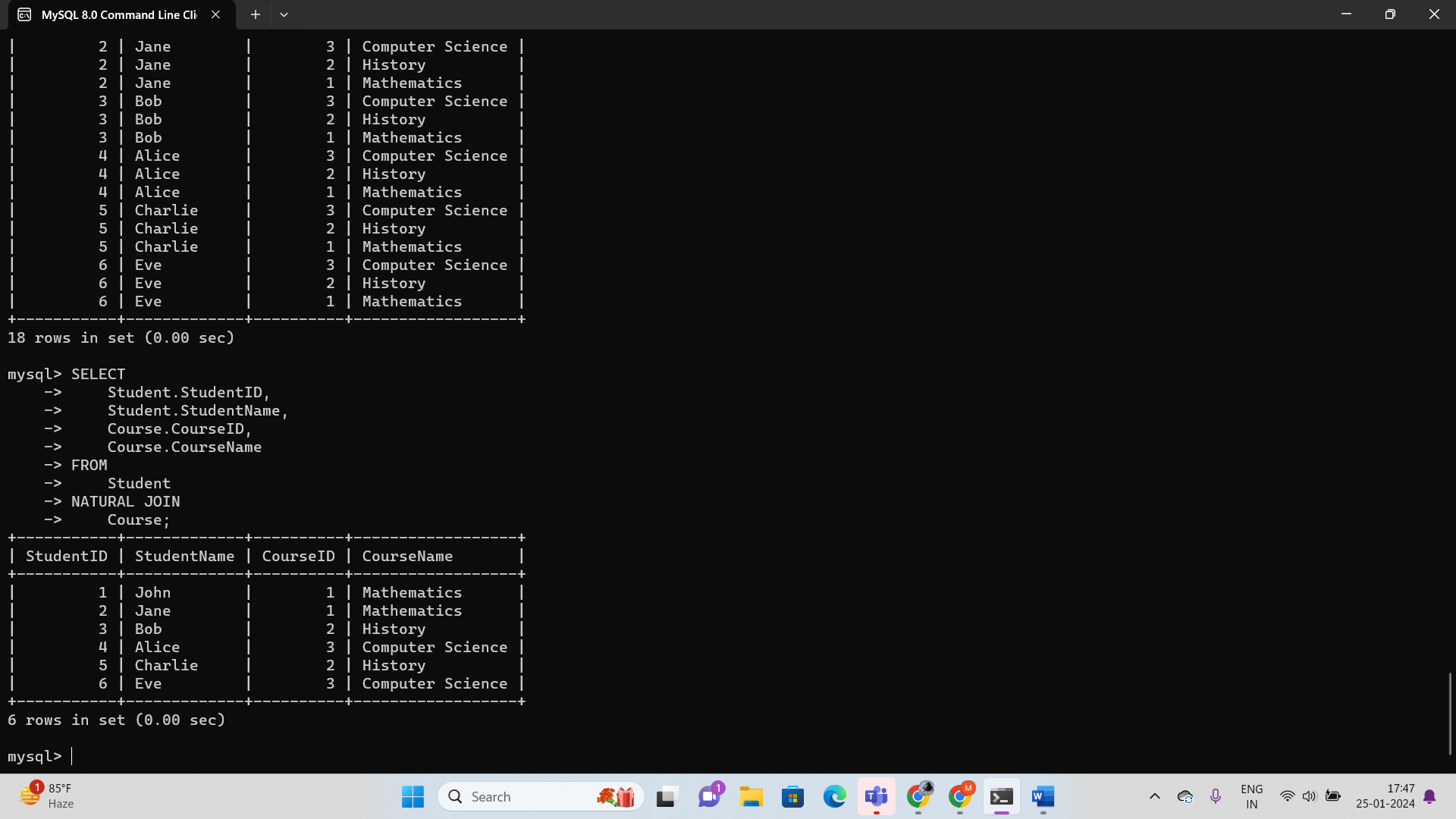
Course.CourseName

FROM

Student

NATURAL JOIN

Course;



In this query:

* **NATURAL JOIN** automatically matches columns with the same name in both tables. In this case, it would match the **CourseID** column.

These examples cover different types of joins in SQL, each serving specific purposes based on the relationships between tables and the desired outcome of the query.