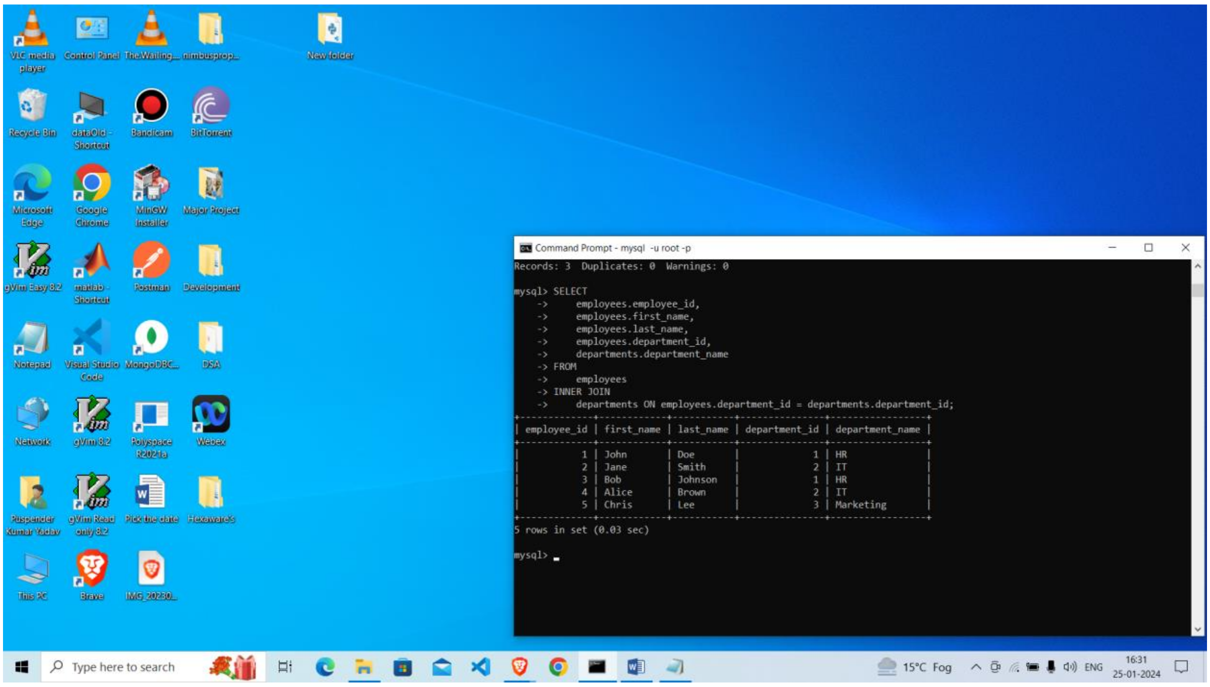
Q.2 Execute all JOINS.

Ans.

INNER JOIN:

An INNER JOIN returns only the rows that have matching values in both tables. Rows from the tables that do not have matching values are excluded from the result set.

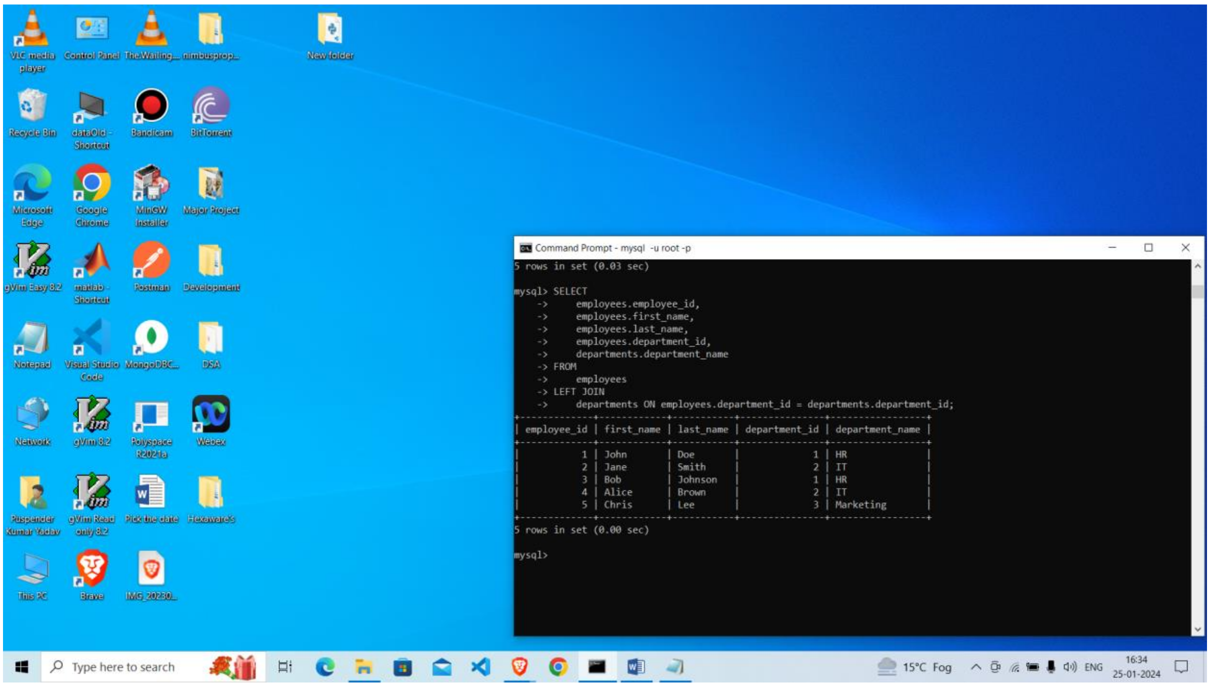


```
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT
    ->     employees.employee_id,
    ->     employees.first_name,
    ->     employees.last_name,
    ->     employees.department_id,
    ->     departments.department_name
    -> FROM
    ->     employees
    -> INNER JOIN
    ->     departments ON employees.department_id = departments.department_id;
+-------------+------------+-----------+---------------+-----------------+
| employee_id | first_name | last_name | department_id | department_name |
+-------------+------------+-----------+---------------+-----------------+
|           1 | John       | Doe       |             1 | HR              |
|           2 | Jane       | Smith     |             2 | IT              |
|           3 | Bob        | Johnson   |             1 | HR              |
|           4 | Alice      | Brown     |             2 | IT              |
|           5 | Chris      | Lee       |             3 | Marketing       |
+-------------+------------+-----------+---------------+-----------------+
5 rows in set (0.03 sec)

mysql> _
```

Explanation: The INNER JOIN returns only the rows where there is a match in both tables based on the specified condition (employees.department_id = departments.department_id).

LEFT JOIN (or LEFT OUTER JOIN):

A LEFT JOIN returns all rows from the left table and the matched rows from the right table. If there is no match, NULL values are returned for columns from the right table.

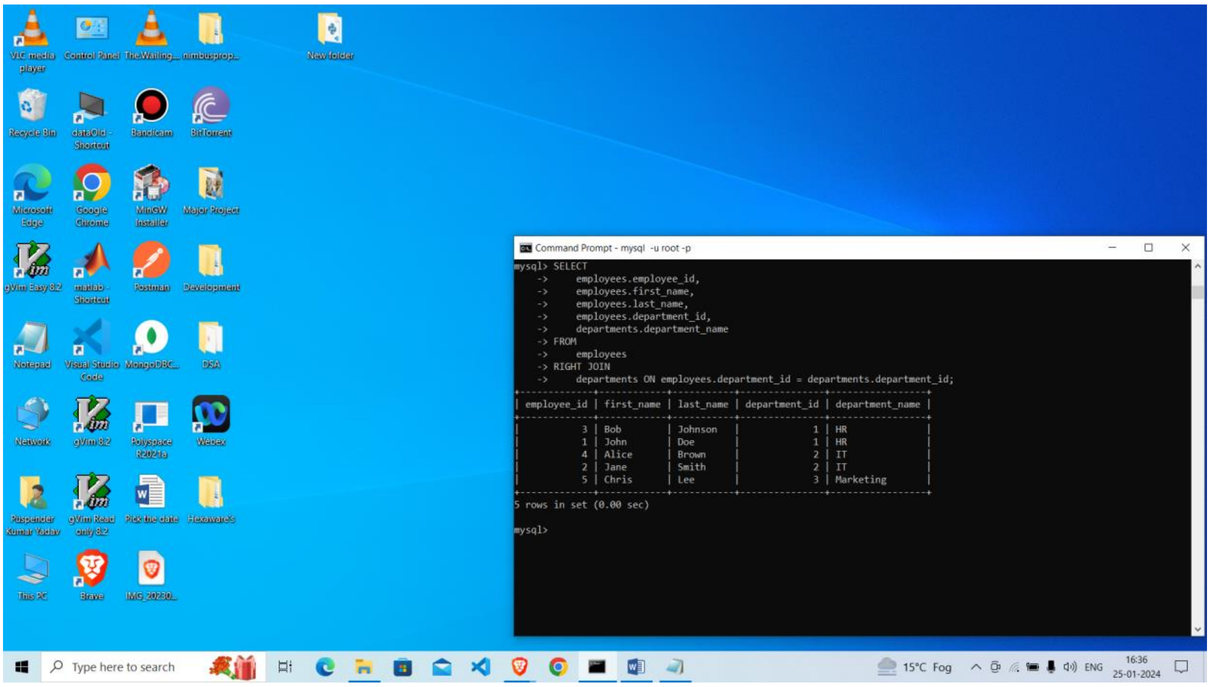


```
5 rows in set (0.03 sec)

mysql> SELECT
    ->     employees.employee_id,
    ->     employees.first_name,
    ->     employees.last_name,
    ->     employees.department_id,
    ->     departments.department_name
    -> FROM
    ->     employees
    -> LEFT JOIN
    ->     departments ON employees.department_id = departments.department_id;
+-------------+------------+-----------+---------------+-----------------+
| employee_id | first_name | last_name | department_id | department_name |
+-------------+------------+-----------+---------------+-----------------+
|           1 | John       | Doe       |             1 | HR              |
|           2 | Jane       | Smith     |             2 | IT              |
|           3 | Bob        | Johnson   |             1 | HR              |
|           4 | Alice      | Brown     |             2 | IT              |
|           5 | Chris      | Lee       |             3 | Marketing       |
+-------------+------------+-----------+---------------+-----------------+
5 rows in set (0.00 sec)

mysql>
```

Explanation: The LEFT JOIN returns all rows from the left table (employees) and the matching rows from the right table (departments). If there is no match, NULL values are returned for the columns from the right table.

RIGHT JOIN (or RIGHT OUTER JOIN):

A RIGHT JOIN is the opposite of the LEFT JOIN. It returns all rows from the right table and the matched rows from the left table. If there is no match, NULL values are returned for columns from the left table.



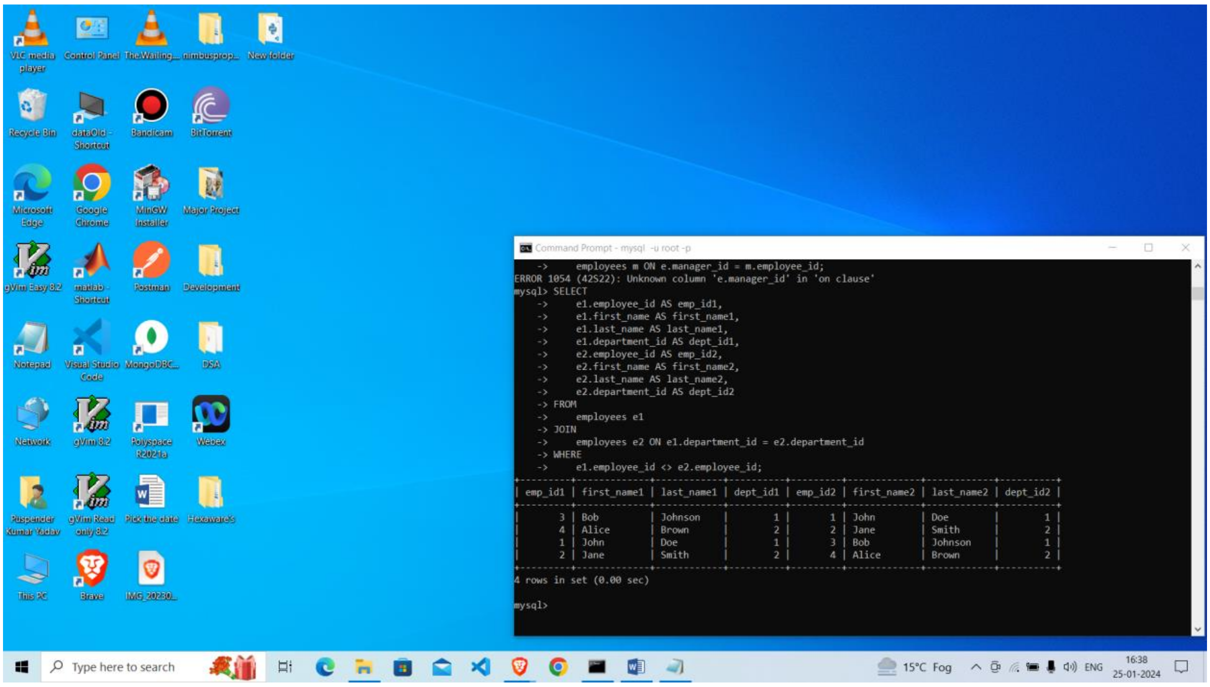
```
mysql> SELECT
    ->     employees.employee_id,
    ->     employees.first_name,
    ->     employees.last_name,
    ->     employees.department_id,
    ->     departments.department_name
    -> FROM
    ->     employees
    -> RIGHT JOIN
    ->     departments ON employees.department_id = departments.department_id;
+-------------+------------+-----------+---------------+-----------------+
| employee_id | first_name | last_name | department_id | department_name |
+-------------+------------+-----------+---------------+-----------------+
|           3 | Bob        | Johnson   |             1 | HR              |
|           1 | John       | Doe       |             1 | HR              |
|           4 | Alice      | Brown     |             2 | IT              |
|           2 | Jane       | Smith     |             2 | IT              |
|           5 | Chris      | Lee       |             3 | Marketing       |
+-------------+------------+-----------+---------------+-----------------+
5 rows in set (0.00 sec)

mysql>
```

Explanation: The RIGHT JOIN returns all rows from the right table (departments) and the matching rows from the left table (employees). If there is no match, NULL values are returned for the columns from the left table.

Self Join:

A self join is a regular join, but it involves joining a table with itself. This is useful when you have a table with hierarchical data or when you want to compare rows within the same table.

Example:



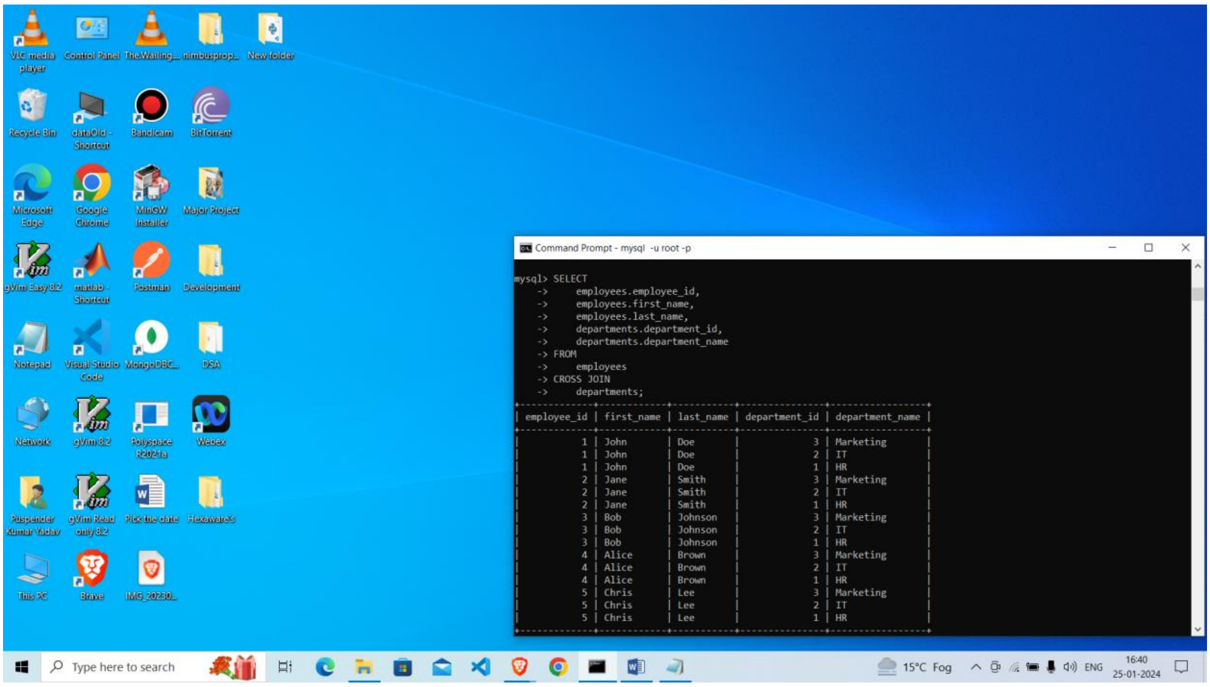
```
    ->     employees m ON e.manager_id = m.employee_id;
ERROR 1054 (42S22): Unknown column 'e.manager_id' in 'on clause'
mysql> SELECT
    ->     e1.employee_id AS emp_id1,
    ->     e1.first_name AS first_name1,
    ->     e1.last_name AS last_name1,
    ->     e1.department_id AS dept_id1,
    ->     e2.employee_id AS emp_id2,
    ->     e2.first_name AS first_name2,
    ->     e2.last_name AS last_name2,
    ->     e2.department_id AS dept_id2
    -> FROM
    ->     employees e1
    -> JOIN
    ->     employees e2 ON e1.department_id = e2.department_id
    -> WHERE
    ->     e1.employee_id <> e2.employee_id;
+---------+-------------+------------+----------+---------+-------------+------------+----------+
| emp_id1 | first_name1 | last_name1 | dept_id1 | emp_id2 | first_name2 | last_name2 | dept_id2 |
+---------+-------------+------------+----------+---------+-------------+------------+----------+
|       3 | Bob         | Johnson    |        1 |       1 | John        | Doe        |        1 |
|       4 | Alice       | Brown      |        2 |       2 | Jane        | Smith      |        2 |
|       1 | John        | Doe        |        1 |       3 | Bob         | Johnson    |        1 |
|       2 | Jane        | Smith      |        2 |       4 | Alice       | Brown      |        2 |
+---------+-------------+------------+----------+---------+-------------+------------+----------+
4 rows in set (0.00 sec)

mysql>
```

Explanation: In this query, we are performing a self join on the employees table based on the department_id. The condition e1.employee_id <> e2.employee_id ensures that we don't match a row with itself. This query retrieves pairs of employees who belong to the same department but are different individuals.

Cross Join:

A cross join returns the Cartesian product of two tables, meaning it combines each row from the first table with every row from the second table. It doesn't require a specific join condition.

Example:



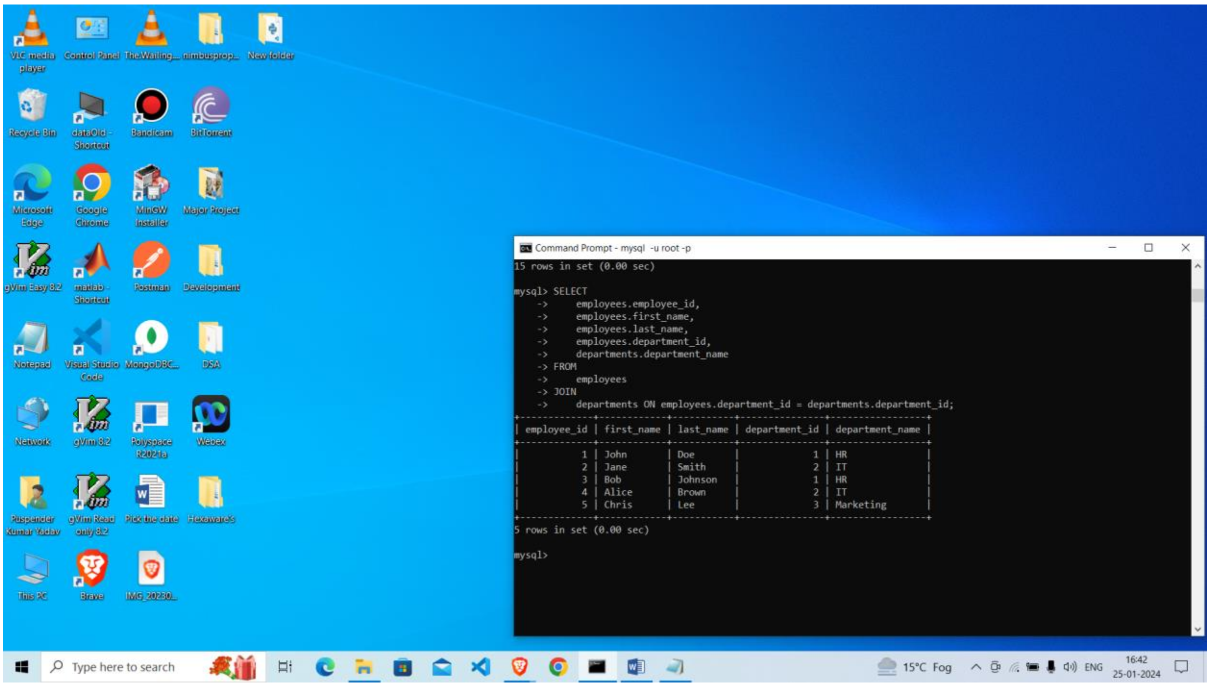
```
mysql> SELECT
    ->     employees.employee_id,
    ->     employees.first_name,
    ->     employees.last_name,
    ->     departments.department_id,
    ->     departments.department_name
    -> FROM
    ->     employees
    -> CROSS JOIN
    ->     departments;
+-------------+------------+-----------+---------------+-----------------+
| employee_id | first_name | last_name | department_id | department_name |
+-------------+------------+-----------+---------------+-----------------+
|           1 | John       | Doe       |             3 | Marketing       |
|           1 | John       | Doe       |             2 | IT              |
|           1 | John       | Doe       |             1 | HR              |
|           2 | Jane       | Smith     |             3 | Marketing       |
|           2 | Jane       | Smith     |             2 | IT              |
|           2 | Jane       | Smith     |             1 | HR              |
|           3 | Bob        | Johnson   |             3 | Marketing       |
|           3 | Bob        | Johnson   |             2 | IT              |
|           3 | Bob        | Johnson   |             1 | HR              |
|           4 | Alice      | Brown     |             3 | Marketing       |
|           4 | Alice      | Brown     |             2 | IT              |
|           4 | Alice      | Brown     |             1 | HR              |
|           5 | Chris      | Lee       |             3 | Marketing       |
|           5 | Chris      | Lee       |             2 | IT              |
|           5 | Chris      | Lee       |             1 | HR              |
+-------------+------------+-----------+---------------+-----------------+
```

Explanation: This query combines every row from the employees table with every row from the departments table, resulting in all possible combinations of employees and departments. Cross joins can lead to a large number of rows, so use them judiciously.

Equi Join:

An equi join is a join where the condition involves equality between columns. It is the most common type of join.

Example:



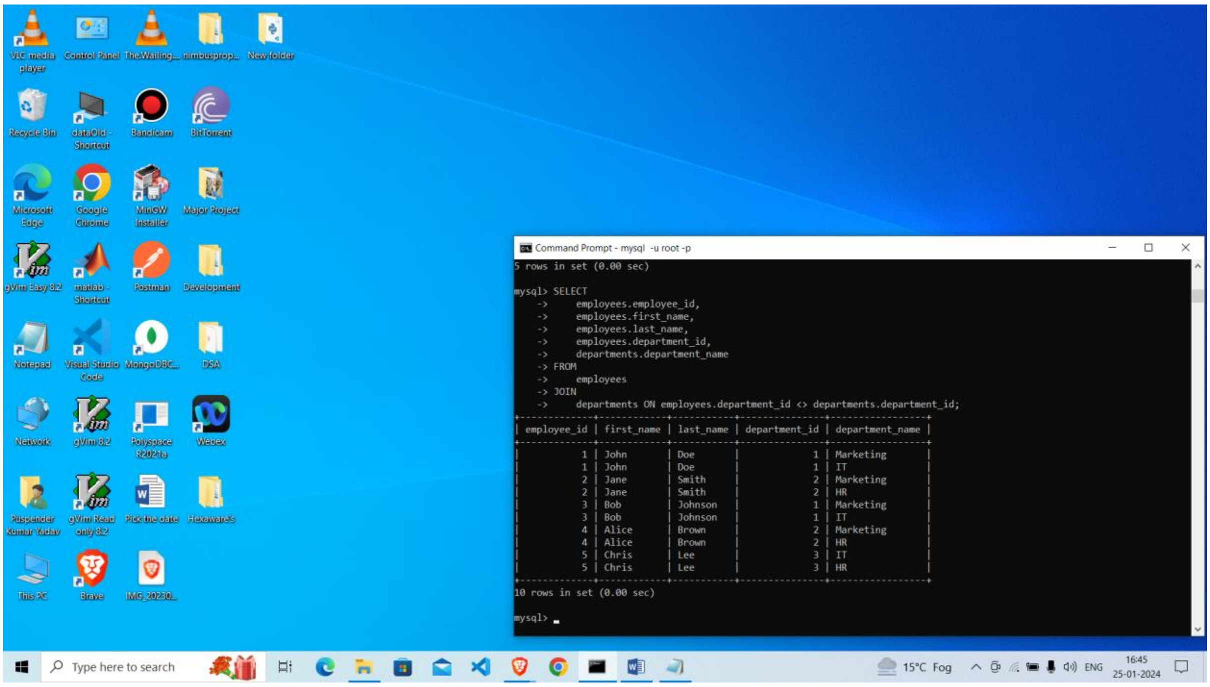
```
15 rows in set (0.00 sec)

mysql> SELECT
    ->     employees.employee_id,
    ->     employees.first_name,
    ->     employees.last_name,
    ->     employees.department_id,
    ->     departments.department_name
    -> FROM
    ->     employees
    -> JOIN
    ->     departments ON employees.department_id = departments.department_id;
+-------------+------------+-----------+---------------+-----------------+
| employee_id | first_name | last_name | department_id | department_name |
+-------------+------------+-----------+---------------+-----------------+
|           1 | John       | Doe       |             1 | HR              |
|           2 | Jane       | Smith     |             2 | IT              |
|           3 | Bob        | Johnson   |             1 | HR              |
|           4 | Alice      | Brown     |             2 | IT              |
|           5 | Chris      | Lee       |             3 | Marketing       |
+-------------+------------+-----------+---------------+-----------------+
5 rows in set (0.00 sec)

mysql>
```

Explanation: In this query, the equi join is based on the equality of employees.department_id and departments.department_id. This retrieves rows where the department IDs match in both tables.

Non-Equi Join:

A non-equi join involves a condition other than equality. It could be inequality, greater than, less than, etc.

Example:



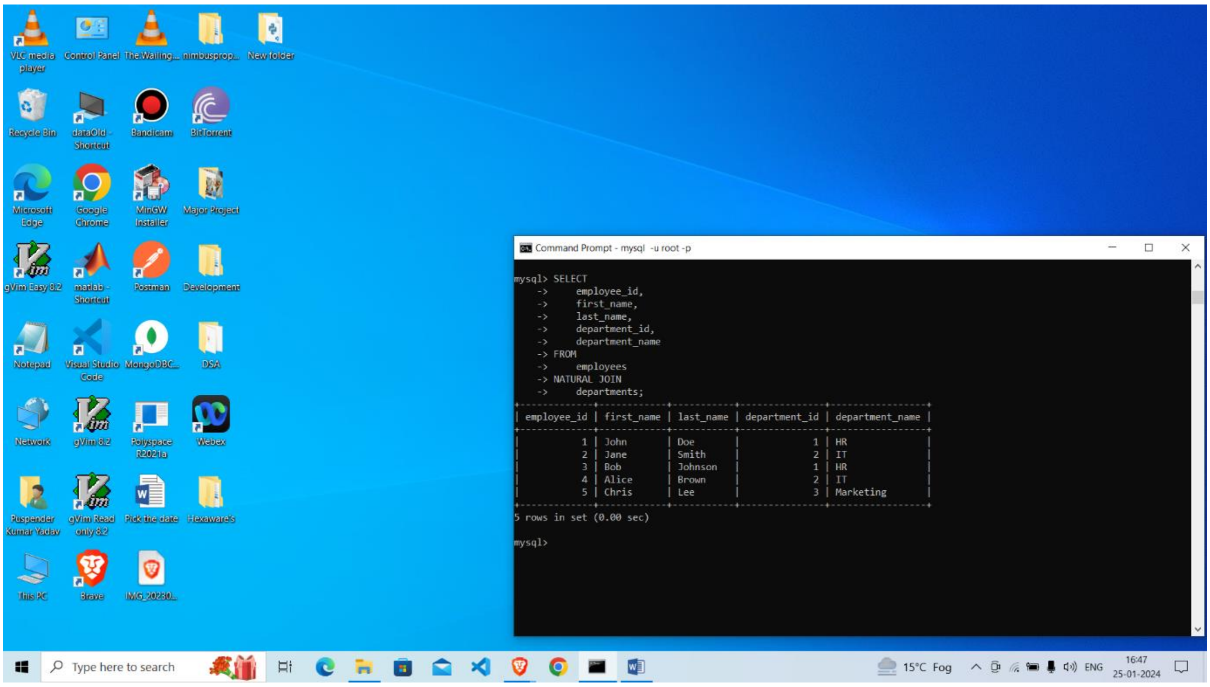
Command Prompt - mysql -u root -p

```
5 rows in set (0.00 sec)

mysql> SELECT
    ->     employees.employee_id,
    ->     employees.first_name,
    ->     employees.last_name,
    ->     employees.department_id,
    ->     departments.department_name
    -> FROM
    ->     employees
    -> JOIN
    ->     departments ON employees.department_id <> departments.department_id;
+-------------+------------+-----------+---------------+-----------------+
| employee_id | first_name | last_name | department_id | department_name |
+-------------+------------+-----------+---------------+-----------------+
|           1 | John       | Doe       |             1 | Marketing       |
|           1 | John       | Doe       |             1 | IT              |
|           2 | Jane       | Smith     |             2 | Marketing       |
|           2 | Jane       | Smith     |             2 | HR              |
|           3 | Bob        | Johnson   |             1 | Marketing       |
|           3 | Bob        | Johnson   |             1 | IT              |
|           4 | Alice      | Brown     |             2 | Marketing       |
|           4 | Alice      | Brown     |             2 | HR              |
|           5 | Chris      | Lee       |             3 | IT              |
|           5 | Chris      | Lee       |             3 | HR              |
+-------------+------------+-----------+---------------+-----------------+
10 rows in set (0.00 sec)

mysql> _
```

Explanation: In this example, the non-equi join is based on the condition employees.department_id <> departments.department_id, which retrieves rows where the department IDs are not equal. This type of join is less common but can be useful in certain scenarios.

Natural Join

A NATURAL JOIN is a type of join that automatically matches columns with the same names in both tables. It eliminates the need to specify the column names explicitly. However, the use of NATURAL JOIN is often discouraged in practice because it can lead to unexpected results if the tables have additional columns with the same names.



```
mysql> SELECT
    ->     employee_id,
    ->     first_name,
    ->     last_name,
    ->     department_id,
    ->     department_name
    -> FROM
    ->     employees
    -> NATURAL JOIN
    ->     departments;
+-------------+------------+-----------+---------------+-----------------+
| employee_id | first_name | last_name | department_id | department_name |
+-------------+------------+-----------+---------------+-----------------+
|           1 | John       | Doe       |             1 | HR              |
|           2 | Jane       | Smith     |             2 | IT              |
|           3 | Bob        | Johnson   |             1 | HR              |
|           4 | Alice      | Brown     |             2 | IT              |
|           5 | Chris      | Lee       |             3 | Marketing       |
+-------------+------------+-----------+---------------+-----------------+
5 rows in set (0.00 sec)

mysql>
```

Explanation: In this query, the NATURAL JOIN automatically matches the columns with the same names (department_id) in both tables (employees and departments). The result includes only the columns with unique names, and the common column (department_id) is included only once.