Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Lab Report
on
"ExaminationResults_DB"

[Code No: COMP 232]

Submitted by

Aayush Karna (18)

Submitted to

Mr. Sanjog Sigdel

Department of Computer Science and Engineering

Submission Date: 31st December, 2024

Link to GitHub Repository: **https://github.com/AayushKarna/dbms**

Table of Contents

Basics

1. **Show all databases:**

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

2. **Create database named ‘examination_result’**

```
mysql> CREATE DATABASE examination_result;
Query OK, 1 row affected (0.01 sec)
```

.

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| examination_result |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

3. **Use examination result database:**

```
mysql> USE examination_result;
Database changed
```

4. **Create ‘students’ table:**

```
mysql> CREATE TABLE students
    -> (
    -> id SERIAL PRIMARY KEY,
    -> name VARCHAR(255),
    -> phone_no VARCHAR(10)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> SHOW TABLES;
+-----------------------------+
| Tables_in_examination_result |
+-----------------------------+
| students                    |
+-----------------------------+
1 row in set (0.00 sec)
```

```
mysql> DESCRIBE students;
+----------+-----------------+------+-----+---------+----------------+
| Field    | Type            | Null | Key | Default | Extra          |
+----------+-----------------+------+-----+---------+----------------+
| id       | bigint unsigned | NO   | PRI | NULL    | auto_increment |
| name     | varchar(255)    | YES  |     | NULL    |                |
| phone_no | varchar(10)     | YES  |     | NULL    |                |
+----------+-----------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

5. **Insert data to the ‘students’ table:**

```
mysql> INSERT INTO students (name, phone_no) VALUES
    -> ('Sita Sharma', '9812345678'),
    -> ('Ram Bahadur', '9812345679'),
    -> ('Gita Dhakal', '9801234567'),
    -> ('Bikash Thapa', '9812345680'),
    -> ('Anita Rai', '9801234568'),
    -> ('Rajesh Shrestha', '9812345681'),
    -> ('Kiran Gurung', '9812345682'),
    -> ('Puja Khadka', '9801234569'),
    -> ('Suman Tamang', '9812345683'),
    -> ('Nisha Adhikari', '9812345684'),
    -> ('Bimal Magar', '9812345685'),
    -> ('Sunita Karki', '9801234570'),
    -> ('Prakash Lama', '9723456789'),
    -> ('Sarita Basnet', '9723456790'),
    -> ('Ramesh Pokharel', '9812345686');
Query OK, 15 rows affected (0.01 sec)
Records: 15  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT * FROM students;
+----+-----------------+------------+
| id | name            | phone_no   |
+----+-----------------+------------+
|  1 | Sita Sharma     | 9812345678 |
|  2 | Ram Bahadur     | 9812345679 |
|  3 | Gita Dhakal     | 9801234567 |
|  4 | Bikash Thapa    | 9812345680 |
|  5 | Anita Rai       | 9801234568 |
|  6 | Rajesh Shrestha | 9812345681 |
|  7 | Kiran Gurung    | 9812345682 |
|  8 | Puja Khadka     | 9801234569 |
|  9 | Suman Tamang    | 9812345683 |
| 10 | Nisha Adhikari  | 9812345684 |
| 11 | Bimal Magar     | 9812345685 |
| 12 | Sunita Karki    | 9801234570 |
| 13 | Prakash Lama    | 9723456789 |
| 14 | Sarita Basnet   | 9723456790 |
| 15 | Ramesh Pokharel | 9812345686 |
+----+-----------------+------------+
15 rows in set (0.00 sec)
```

6. **Create ‘courses’ table:**

```
mysql> CREATE TABLE courses
    -> (
    -> id SERIAL PRIMARY KEY,
    -> name VARCHAR(255),
    -> course_code VARCHAR(10),
    -> credit_hour INTEGER
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> DESCRIBE courses
    -> ;
+-------------+-----------------+------+-----+---------+----------------+
| Field       | Type            | Null | Key | Default | Extra          |
+-------------+-----------------+------+-----+---------+----------------+
| id          | bigint unsigned | NO   | PRI | NULL    | auto_increment |
| name        | varchar(255)    | YES  |     | NULL    |                |
| course_code | varchar(10)     | YES  |     | NULL    |                |
| credit_hour | int             | YES  |     | NULL    |                |
+-------------+-----------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

7. **Insert courses:**

```
mysql> INSERT INTO courses (name, course_code, credit_hour) VALUES
    -> ('Introduction to Programming', 'CS101', 3),
    -> ('Data Structures and Algorithms', 'CS201', 4),
    -> ('Database Management Systems', 'CS301', 3),
    -> ('Computer Networks', 'CS401', 3),
    -> ('Operating Systems', 'CS302', 4),
    -> ('Software Engineering', 'CS402', 3),
    -> ('Artificial Intelligence', 'CS403', 3),
    -> ('Web Development', 'CS202', 3),
    -> ('Mobile Application Development', 'CS404', 3),
    -> ('Discrete Mathematics', 'MATH201', 3),
    -> ('Computer Graphics', 'CS405', 3),
    -> ('Machine Learning', 'CS406', 4),
    -> ('Cloud Computing', 'CS407', 3),
    -> ('Theory of Computation', 'CS303', 3),
    -> ('Cyber Security', 'CS408', 3),
    -> ('Linear Algebra', 'MATH101', 3),
    -> ('Calculus', 'MATH102', 4),
    -> ('Combinatorics', 'MATH301', 3);
Query OK, 18 rows affected (0.01 sec)
Records: 18  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT * FROM courses;
+----+-------------------------------+-------------+-------------+
| id | name                          | course_code | credit_hour |
+----+-------------------------------+-------------+-------------+
|  1 | Introduction to Programming   | CS101       |           3 |
|  2 | Data Structures and Algorithms | CS201       |           4 |
|  3 | Database Management Systems   | CS301       |           3 |
|  4 | Computer Networks             | CS401       |           3 |
|  5 | Operating Systems             | CS302       |           4 |
|  6 | Software Engineering          | CS402       |           3 |
|  7 | Artificial Intelligence       | CS403       |           3 |
|  8 | Web Development               | CS202       |           3 |
|  9 | Mobile Application Development | CS404       |           3 |
| 10 | Discrete Mathematics          | MATH201     |           3 |
| 11 | Computer Graphics             | CS405       |           3 |
| 12 | Machine Learning              | CS406       |           4 |
| 13 | Cloud Computing               | CS407       |           3 |
| 14 | Theory of Computation         | CS303       |           3 |
| 15 | Cyber Security                | CS408       |           3 |
| 16 | Linear Algebra                | MATH101     |           3 |
| 17 | Calculus                      | MATH102     |           4 |
| 18 | Combinatorics                 | MATH301     |           3 |
+----+-------------------------------+-------------+-------------+
18 rows in set (0.00 sec)
```

Relational Algebra

Create database with at least two tables and make sure that the tables are associated with foreign key:

1. **Create ‘customers’ table:**

```
mysql> CREATE TABLE customers (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(255) NOT NULL, email VARCHAR(255
) UNIQUE NOT NULL, city VARCHAR(50));
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> DESCRIBE customers
    -> ;;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | int          | NO   | PRI | NULL    | auto_increment |
| name  | varchar(255) | NO   |     | NULL    |                |
| email | varchar(255) | NO   | UNI | NULL    |                |
| city  | varchar(50)  | YES  |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

2. Insert data to the ‘customers’ table:

```
mysql> INSERT INTO Customers (name, email, city) VALUES
    -> ('Ram Bahadur Thapa', 'ram.thapa@gmail.com', 'Kathmandu'),
    -> ('Sita Kumari Sharma', 'sita.sharma@yahoo.com', 'Pokhara'),
    -> ('Bishnu Prasad Poudel', 'bishnu.poudel@hotmail.com', 'Biratnagar'),
    -> ('Gita Devi Gautam', 'gita.gautam@gmail.com', 'Lalitpur'),
    -> ('Hari Krishna Joshi', 'hari.joshi@gmail.com', 'Bhaktapur'),
    -> ('Ramesh Bhandari', 'ramesh.bhandari@gmail.com', 'Chitwan'),
    -> ('Mina Khadka', 'mina.khadka@gmail.com', 'Butwal'),
    -> ('Santosh Adhikari', 'santosh.adhikari@gmail.com', 'Dharan'),
    -> ('Kamal Shrestha', 'kamal.shrestha@gmail.com', 'Hetauda'),
    -> ('Sunita Rai', 'sunita.rai@gmail.com', 'Birgunj');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT * FROM customers;
+----+---------------------+---------------------------+------------+
| id | name                | email                     | city       |
+----+---------------------+---------------------------+------------+
|  1 | Ram Bahadur Thapa   | ram.thapa@gmail.com       | Kathmandu  |
|  2 | Sita Kumari Sharma  | sita.sharma@yahoo.com     | Pokhara    |
|  3 | Bishnu Prasad Poudel | bishnu.poudel@hotmail.com | Biratnagar |
|  4 | Gita Devi Gautam    | gita.gautam@gmail.com     | Lalitpur   |
|  5 | Hari Krishna Joshi  | hari.joshi@gmail.com      | Bhaktapur  |
|  6 | Ramesh Bhandari     | ramesh.bhandari@gmail.com | Chitwan    |
|  7 | Mina Khadka         | mina.khadka@gmail.com     | Butwal     |
|  8 | Santosh Adhikari    | santosh.adhikari@gmail.com | Dharan     |
|  9 | Kamal Shrestha      | kamal.shrestha@gmail.com  | Hetauda    |
| 10 | Sunita Rai          | sunita.rai@gmail.com      | Birgunj    |
+----+---------------------+---------------------------+------------+
10 rows in set (0.00 sec)
```

3. Create ‘orders table:

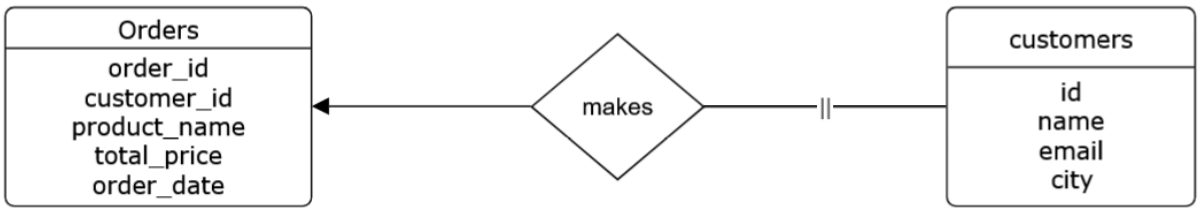
```
mysql> CREATE TABLE Orders (
    ->     order_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     customer_id INT NOT NULL,
    ->     product_name VARCHAR(100) NOT NULL,
    ->     total_price DECIMAL(10, 2) NOT NULL,
    ->     order_date DATE NOT NULL,
    ->     FOREIGN KEY (customer_id) REFERENCES customers(id)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

4. **Insert orders:**

```
mysql> INSERT INTO Orders (customer_id, product_name, total_price, order_date) VALUES
    -> (1, 'Laptop', 75000.00, '2024-11-01'),
    -> (2, 'Smartphone', 25000.00, '2024-11-02'),
    -> (3, 'Headphones', 5000.00, '2024-11-03'),
    -> (4, 'Tablet', 30000.00, '2024-11-04'),
    -> (5, 'Smartwatch', 15000.00, '2024-11-05'),
    -> (6, 'Camera', 45000.00, '2024-11-06'),
    -> (7, 'Gaming Console', 40000.00, '2024-11-07'),
    -> (8, 'Keyboard', 2000.00, '2024-11-08'),
    -> (9, 'Monitor', 20000.00, '2024-11-09'),
    -> (10, 'Printer', 8000.00, '2024-11-10');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT * FROM Orders;
+----------+-------------+----------------+-------------+------------+
| order_id | customer_id | product_name   | total_price | order_date |
+----------+-------------+----------------+-------------+------------+
|        1 |           1 | Laptop         |    75000.00 | 2024-11-01 |
|        2 |           2 | Smartphone     |    25000.00 | 2024-11-02 |
|        3 |           3 | Headphones     |     5000.00 | 2024-11-03 |
|        4 |           4 | Tablet         |    30000.00 | 2024-11-04 |
|        5 |           5 | Smartwatch     |    15000.00 | 2024-11-05 |
|        6 |           6 | Camera         |    45000.00 | 2024-11-06 |
|        7 |           7 | Gaming Console |    40000.00 | 2024-11-07 |
|        8 |           8 | Keyboard       |     2000.00 | 2024-11-08 |
|        9 |           9 | Monitor        |    20000.00 | 2024-11-09 |
|       10 |          10 | Printer        |     8000.00 | 2024-11-10 |
+----------+-------------+----------------+-------------+------------+
10 rows in set (0.00 sec)
```

Relationship model between ‘customers’ and ‘Orders’ table:



One to many: one customer can make many orders.

Relational Algebraic Operations:

1. **Selection (σ):**

```
mysql> SELECT * FROM Orders WHERE total_price >= 20000;
+----------+-------------+---------------+-------------+------------+
| order_id | customer_id | product_name  | total_price | order_date |
+----------+-------------+---------------+-------------+------------+
|        1 |           1 | Laptop        |    75000.00 | 2024-11-01 |
|        2 |           2 | Smartphone    |    25000.00 | 2024-11-02 |
|        4 |           4 | Tablet        |    30000.00 | 2024-11-04 |
|        6 |           6 | Camera        |    45000.00 | 2024-11-06 |
|        7 |           7 | Gaming Console |    40000.00 | 2024-11-07 |
|        9 |           9 | Monitor       |    20000.00 | 2024-11-09 |
+----------+-------------+---------------+-------------+------------+
6 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM customers WHERE city = 'Kathmandu';
+----+------------------+--------------------+-----------+
| id | name             | email              | city      |
+----+------------------+--------------------+-----------+
|  1 | Ram Bahadur Thapa | ram.thapa@gmail.com | Kathmandu |
+----+------------------+--------------------+-----------+
1 row in set (0.00 sec)
```

2. **Projection (π):**

```
mysql> SELECT name, email FROM customers;
+---------------------+---------------------------+
| name                | email                     |
+---------------------+---------------------------+
| Ram Bahadur Thapa   | ram.thapa@gmail.com       |
| Sita Kumari Sharma  | sita.sharma@yahoo.com     |
| Bishnu Prasad Poudel | bishnu.poudel@hotmail.com |
| Gita Devi Gautam    | gita.gautam@gmail.com     |
| Hari Krishna Joshi  | hari.joshi@gmail.com      |
| Ramesh Bhandari     | ramesh.bhandari@gmail.com |
| Mina Khadka         | mina.khadka@gmail.com     |
| Santosh Adhikari    | santosh.adhikari@gmail.com |
| Kamal Shrestha      | kamal.shrestha@gmail.com  |
| Sunita Rai          | sunita.rai@gmail.com      |
+---------------------+---------------------------+
10 rows in set (0.00 sec)
```

```
mysql> SELECT product_name, total_price
    -> FROM Orders;
+----------------+-------------+
| product_name   | total_price |
+----------------+-------------+
| Laptop         |    75000.00 |
| Smartphone     |    25000.00 |
| Headphones     |     5000.00 |
| Tablet         |    30000.00 |
| Smartwatch     |    15000.00 |
| Camera         |    45000.00 |
| Gaming Console |    40000.00 |
| Keyboard       |     2000.00 |
| Monitor        |    20000.00 |
| Printer        |     8000.00 |
+----------------+-------------+
10 rows in set (0.00 sec)
```

3. **Cartesian Product (X):**

a) Union

```
mysql> SELECT id FROM customers
    -> UNION
    -> SELECT customer_id FROM Orders;
+----+
| id |
+----+
|  3 |
|  4 |
|  5 |
|  9 |
|  7 |
|  1 |
|  6 |
|  8 |
|  2 |
| 10 |
+----+
10 rows in set (0.00 sec)
```

b) Intersection

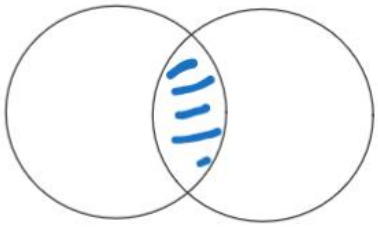
```
mysql> SELECT id FROM customers
    -> INTERSECT
    -> SELECT customer_id FROM Orders;
+----+
| id |
+----+
|  3 |
|  4 |
|  5 |
|  9 |
|  7 |
|  1 |
|  6 |
|  8 |
|  2 |
| 10 |
+----+
10 rows in set (0.00 sec)
```

c) Set Difference (EXCEPT)

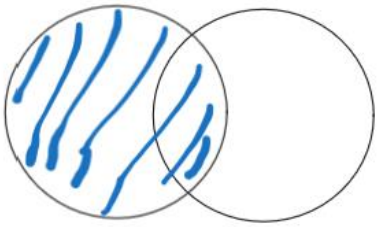
```
mysql> SELECT id FROM customers
    -> EXCEPT
    -> SELECT customer_id FROM Orders;
Empty set (0.00 sec)
```

Currently every customer has placed at least one order so the set difference is empty, otherwise it would show customer that have not made an order.

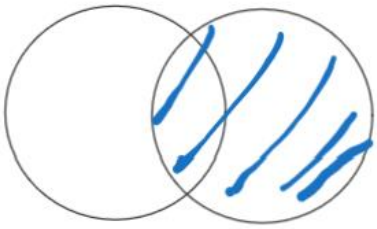
Joins



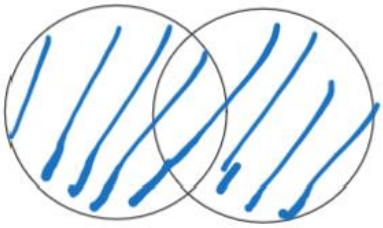
Inner Join



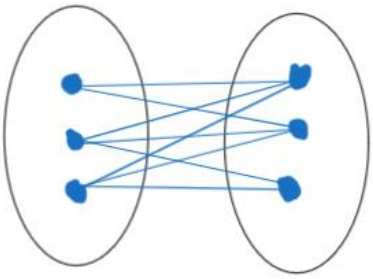
Left Join



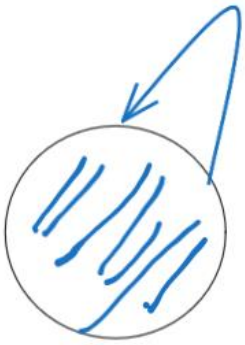
Right Join



Full Join



Cross Join



Self Join

1. **Create Tables**

```
mysql> CREATE TABLE Employees (
    ->     EmployeeID INT AUTO_INCREMENT PRIMARY KEY,
    ->     Name VARCHAR(100),
    ->     DepartmentID INT
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE Departments (
    ->     DepartmentID INT AUTO_INCREMENT PRIMARY KEY,
    ->     DepartmentName VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> DESCRIBE Employees;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| EmployeeID   | int          | NO   | PRI | NULL    | auto_increment |
| Name         | varchar(100) | YES  |     | NULL    |                |
| DepartmentID | int          | YES  |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)

mysql> DESCRIBE Departments;
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| DepartmentID   | int          | NO   | PRI | NULL    | auto_increment |
| DepartmentName | varchar(100) | YES  |     | NULL    |                |
+----------------+--------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)
```

2. **Insert Data into tables**

```
mysql> INSERT INTO Employees (Name, DepartmentID) VALUES
    -> ('Alice', 1),
    -> ('Bob', 2),
    -> ('Charlie', 3),
    -> ('Diana', NULL),
    -> ('Eve', 1);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql>
mysql> INSERT INTO Departments (DepartmentName) VALUES
    -> ('HR'),
    -> ('Finance'),
    -> ('Engineering'),
    -> ('Marketing');
Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Employees;
+------------+---------+--------------+
| EmployeeID | Name    | DepartmentID |
+------------+---------+--------------+
|          1 | Alice   |            1 |
|          2 | Bob     |            2 |
|          3 | Charlie |            3 |
|          4 | Diana   |         NULL |
|          5 | Eve     |            1 |
+------------+---------+--------------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM Departments;
+--------------+----------------+
| DepartmentID | DepartmentName |
+--------------+----------------+
|            1 | HR             |
|            2 | Finance        |
|            3 | Engineering    |
|            4 | Marketing      |
+--------------+----------------+
4 rows in set (0.00 sec)
```

3. **Performing different types of JOINs**

a. **Inner JOIN:**
Retrieves records that have matching values in both tables.

```
mysql> SELECT Employees.Name, Departments.DepartmentName
    -> FROM Employees
    -> INNER JOIN Departments
    -> ON Employees.DepartmentID = Departments.DepartmentID;
+---------+----------------+
| Name    | DepartmentName |
+---------+----------------+
| Alice   | HR             |
| Bob     | Finance        |
| Charlie | Engineering    |
| Eve     | HR             |
+---------+----------------+
4 rows in set (0.00 sec)
```

b. **Left Join:**
Retrieves all records from the left table, and the matched records from the right table. Unmatched row will show NULL for columns from the right table.

```
mysql> SELECT Employees.Name, Departments.DepartmentName
    -> FROM Employees
    -> LEFT JOIN Departments
    -> ON Employees.DepartmentID = Departments.DepartmentID;
+---------+----------------+
| Name    | DepartmentName |
+---------+----------------+
| Alice   | HR             |
| Bob     | Finance        |
| Charlie | Engineering    |
| Diana   | NULL           |
| Eve     | HR             |
+---------+----------------+
5 rows in set (0.00 sec)
```

c. **Right Join:**
Retrieves all records from the right table, and the matched records from the left

table. Unmatched rows will show NULL for columns from the left table.

```
mysql> SELECT Employees.Name, Departments.DepartmentName
    -> FROM Employees
    -> RIGHT JOIN Departments
    -> ON Employees.DepartmentID = Departments.DepartmentID;
+---------+----------------+
| Name    | DepartmentName |
+---------+----------------+
| Eve     | HR             |
| Alice   | HR             |
| Bob     | Finance        |
| Charlie | Engineering    |
| NULL    | Marketing      |
+---------+----------------+
5 rows in set (0.00 sec)
```

d. Full Outer Join:

Retrieves all records when there is a match in either table (not directly supported in MySQL; achieved using a UNION).

```
mysql> SELECT Employees.Name, Departments.DepartmentName
    -> FROM Employees
    -> LEFT JOIN Departments
    -> ON Employees.DepartmentID = Departments.DepartmentID
    -> UNION
    -> SELECT Employees.Name, Departments.DepartmentName
    -> FROM Employees
    -> RIGHT JOIN Departments
    -> ON Employees.DepartmentID = Departments.DepartmentID;
+---------+----------------+
| Name    | DepartmentName |
+---------+----------------+
| Alice   | HR             |
| Bob     | Finance        |
| Charlie | Engineering    |
| Diana   | NULL           |
| Eve     | HR             |
| NULL    | Marketing      |
+---------+----------------+
6 rows in set (0.00 sec)
```

e. Cross Join:

Produces the Cartesian product of the two tables (every combination of rows).

```
mysql> SELECT Employees.Name, Departments.DepartmentName
    -> FROM Employees
    -> CROSS JOIN Departments;
+---------+----------------+
| Name    | DepartmentName |
+---------+----------------+
| Alice   | Marketing      |
| Alice   | Engineering    |
| Alice   | Finance        |
| Alice   | HR             |
| Bob     | Marketing      |
| Bob     | Engineering    |
| Bob     | Finance        |
| Bob     | HR             |
| Charlie | Marketing      |
| Charlie | Engineering    |
| Charlie | Finance        |
| Charlie | HR             |
| Diana   | Marketing      |
| Diana   | Engineering    |
| Diana   | Finance        |
| Diana   | HR             |
| Eve     | Marketing      |
| Eve     | Engineering    |
| Eve     | Finance        |
| Eve     | HR             |
+---------+----------------+
20 rows in set (0.00 sec)
```

f. **Self Join:**

Joins a table to itself (example: finding pairs of employees in the same

department)

```
mysql> SELECT A.Name AS Employee1, B.Name AS Employee2
    -> FROM Employees A
    -> INNER JOIN Employees B
    -> ON A.DepartmentID = B.DepartmentID
    -> WHERE A.EmployeeID < B.EmployeeID;
+-----------+-----------+
| Employee1 | Employee2 |
+-----------+-----------+
| Alice     | Eve       |
+-----------+-----------+
1 row in set (0.00 sec)
```

Normalization

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing a database into two or more tables and defining relationships between them to eliminate data anomalies.

1. First Normal Form (1NF)

Drawback in Non-1NF: A column contains multiple values (non-atomic values). This makes querying difficult.

```
mysql> CREATE TABLE Student (
    ->     StudentID INT PRIMARY KEY,
    ->     Name VARCHAR(50),
    ->     Subjects VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> INSERT INTO Student (StudentID, Name, Subjects) VALUES (1, 'John', 'Math,Scienc');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Student;
+-----------+------+-------------+
| StudentID | Name | Subjects    |
+-----------+------+-------------+
|         1 | John | Math,Scienc |
+-----------+------+-------------+
1 row in set (0.00 sec)
```

Transformed to 1NF:

Each column contains only atomic values.

```
mysql> CREATE TABLE StudentSubjects (
    ->     StudentID INT,
    ->     Name VARCHAR(50),
    ->     Subject VARCHAR(50),
    ->     PRIMARY KEY (StudentID, Subject)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO Student (StudentID, Name, Subject) VALUES (1, 'John', 'Math'), (1, 'John', 'Science');
ERROR 1054 (42S22): Unknown column 'Subject' in 'field list'
mysql> INSERT INTO StudentSubjects (StudentID, Name, Subject) VALUES (1, 'John', 'Math'), (1, 'John', 'Science'
);
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM StudentSubjects;
+-----------+------+---------+
| StudentID | Name | Subject |
+-----------+------+---------+
|         1 | John | Math    |
|         1 | John | Science |
+-----------+------+---------+
2 rows in set (0.00 sec)
```

2. Second Normal Form (2NF)

Drawback in 1NF: Partial dependencies exist – non-key attribute depends on part of composite key, leading to redundancy.

```
mysql> INSERT INTO Enrollment (StudentID, CourseID, CourseName, Grade) VALUES (1, 101, 'Mathematics', 'A'),
    -> (2, 102, 'Physics', 'B');
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Enrollment;
+-----------+----------+-------------+-------+
| StudentID | CourseID | CourseName  | Grade |
+-----------+----------+-------------+-------+
|         1 |      101 | Mathematics | A     |
|         2 |      102 | Physics     | B     |
+-----------+----------+-------------+-------+
2 rows in set (0.00 sec)
```

Transformed to 2NF:

Split the table to remove partial dependency.

```
mysql> SELECT * FROM Students;
+-----------+------+-------+------------+
| StudentId | Name | Batch | Department |
+-----------+------+-------+------------+
|         1 | John |  2022 | DoCSE      |
+-----------+------+-------+------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM Courses;
ERROR 1146 (42S02): Table 'normalization_practice.courses' doesn't exist
mysql> SELECT * FROM Course;
+----------+-------------+
| CourseID | CourseName  |
+----------+-------------+
|      101 | Mathematics |
|      102 | Physics     |
+----------+-------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM StudentCourse;
+-----------+----------+-------+
| StudentID | CourseID | Grade |
+-----------+----------+-------+
|         1 |      101 | A     |
|         1 |      102 | B     |
+-----------+----------+-------+
2 rows in set (0.00 sec)
```

3. Third Normal Form (3NF)

Drawback in 2NF: Transitive dependency – non-key attribute depend on other non-key attributes.

```
mysql> SELECT * FROM Employee;
+-------+---------+--------+----------+
| EmpID | EmpName | DeptID | DeptName |
+-------+---------+--------+----------+
|     1 | Alice   |     10 | HR       |
|     2 | Bob     |     20 | IT       |
+-------+---------+--------+----------+
2 rows in set (0.00 sec)
```

Transformed to 2NF:

Remove transitive dependency by splitting the table.

```
mysql> SELECT * FROM Employee;
+-------+---------+--------+
| EmpID | EmpName | DeptID |
+-------+---------+--------+
|     1 | Alice   |     10 |
|     2 | Bob     |     20 |
+-------+---------+--------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM Department;
+--------+----------+
| DeptID | DeptName |
+--------+----------+
|     10 | HR       |
|     20 | IT       |
+--------+----------+
2 rows in set (0.00 sec)
```

4. Boyce-Codd Normal Form (BCNF)

Drawback in 3NF: Dependency anomaly – non-superkey attributes can determine other attributes.

```
mysql> SELECT * FROM CourseInstructor;
+------------+------------+-------------+
| CourseName | Instructor | Schedule    |
+------------+------------+-------------+
| Math       | Dr. Smith  | Monday 9 AM |
+------------+------------+-------------+
1 row in set (0.00 sec)
```

Instructor can uniquely determine Schedule, but it is not a superkey.

Transformed to BCNF:

Ensure all determinants are superkey.

```
mysql> SELECT * FROM CourseInstructor;
+------------+------------+
| CourseName | Instructor |
+------------+------------+
| Math       | Dr. Smith  |
+------------+------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM InstructorSchedule;
+------------+-------------+
| Instructor | Schedule    |
+------------+-------------+
| Dr. Smith  | Monday 9 AM |
+------------+-------------+
1 row in set (0.00 sec)
```

5. Fourth Normal Form (4NF)

Drawback in BCNF: Multi-valued dependencies lead to redundancy when two indepented attributes are stored together.

```
mysql> SELECT * FROM EmployeeSkills;
+-------+-------------+----------+
| EmpID | Skill       | Language |
+-------+-------------+----------+
|     1 | Programming | English  |
|     2 | Programming | French   |
+-------+-------------+----------+
2 rows in set (0.00 sec)
```

Here, Skill and Language are independent but stored together.

Transformed to 4NF:

Separate multi-valued dependencies.

```
mysql> SELECT * FROM EmployeeSkill;
+-------+-------------+
| EmpID | Skill       |
+-------+-------------+
|     1 | Programming |
+-------+-------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM EmployeeLanguage;
+-------+----------+
| EmpID | Language |
+-------+----------+
|     1 | English  |
|     1 | French   |
+-------+----------+
```

6. Fifth Normal Form (5NF)

Drawback in 4NF: Join dependency – redundancy exists if tables are not properly decomposed.

```
mysql> SELECT * FROM ProjectAssignment;
+-----------+-------+--------+
| ProjectID | EmpID | RoleID |
+-----------+-------+--------+
|         1 |     1 |    101 |
+-----------+-------+--------+
1 row in set (0.00 sec)
```

Redundancy can occur if multiple independent relationships exist.

Transformed to 5NF:

Decompose to eliminate join dependencies.

```
mysql> SELECT * FROM ProjectROle;
+-----------+--------+
| ProjectID | RoleID |
+-----------+--------+
|         1 |    101 |
+-----------+--------+
1 row in set (0.00 sec)

mysql> SELECT * FROM EmployeeProject;
+-------+-----------+
| EmpID | ProjectID |
+-------+-----------+
|     1 |         1 |
+-------+-----------+
1 row in set (0.00 sec)
```

Transaction, Rollback and Commit

Setup:

1. Create the customers and accounts table

```
mysql> CREATE TABLE customers (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     name VARCHAR(255) NOT NULL
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> CREATE TABLE accounts (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     customer_id INT,
    ->     balance DECIMAL(10, 2) DEFAULT 0,
    ->     FOREIGN KEY (customer_id) REFERENCES customers(id)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> DESCRIBE customers;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | int          | NO   | PRI | NULL    | auto_increment |
| name  | varchar(255) | NO   |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)

mysql> DESCRIBE accounts;
+-------------+---------------+------+-----+---------+----------------+
| Field       | Type          | Null | Key | Default | Extra          |
+-------------+---------------+------+-----+---------+----------------+
| id          | int           | NO   | PRI | NULL    | auto_increment |
| customer_id | int           | YES  | MUL | NULL    |                |
| balance     | decimal(10,2) | YES  |     | 0.00    |                |
+-------------+---------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

2. Insert sample data:

```
mysql> SELECT * FROM customers;
+----+--------+
| id | name   |
+----+--------+
|  1 | User A |
|  2 | User B |
|  3 | User C |
+----+--------+
3 rows in set (0.00 sec)

mysql> INSERT INTO accounts (customer_id, balance) VALUES
    -> (1, 1000),
    -> (2, 2000),
    -> (3, 1500);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM accounts;
+----+-------------+---------+
| id | customer_id | balance |
+----+-------------+---------+
|  1 |           1 | 1000.00 |
|  2 |           2 | 2000.00 |
|  3 |           3 | 1500.00 |
+----+-------------+---------+
3 rows in set (0.00 sec)
```

3. Turn off autocommit

```
mysql> SET autocommit = 0;
Query OK, 0 rows affected (0.00 sec)
```

Transaction With Rollback and Commit with exception handling

```
mysql> DELIMITER $$
mysql>
mysql> CREATE PROCEDURE `transaction_error_case`(
    ->     IN sender_id INT,
    ->     IN receiver_id INT,
    ->     IN transfer_amount DECIMAL(10,2)
    -> )
    -> BEGIN
    ->     -- Declare variables first
    ->     DECLARE sender_balance DECIMAL(10,2);
    ->
    ->     -- Declare exit handler for SQL exceptions
    ->     DECLARE EXIT HANDLER FOR SQLEXCEPTION
    ->     BEGIN
    ->         -- Rollback the transaction on error
    ->         ROLLBACK;
    ->         SELECT 'Transaction failed, rollback executed' AS message;
    ->     END;
    ->
    ->     -- Start transaction
    ->     START TRANSACTION;
    ->
    ->     -- Check if the sender exists
    ->     SELECT balance INTO sender_balance
    ->     FROM accounts
    ->     WHERE customer_id = sender_id;
    ->
    ->     IF sender_balance IS NULL THEN
    ->         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Sender does not exist';
    ->     END IF;
    ->
    ->     -- Check if the sender has sufficient balance
    ->     IF sender_balance < transfer_amount THEN
    ->         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Insufficient balance';
    ->     END IF;
    ->
    ->     -- Check if the receiver exists
    ->     IF NOT EXISTS (SELECT 1 FROM accounts WHERE customer_id = receiver_id) THEN
    ->         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Receiver does not exist';
    ->     END IF;
    ->
    ->     -- Deduct the transfer amount from the sender
    ->     UPDATE accounts
    ->     SET balance = balance - transfer_amount
    ->     WHERE customer_id = sender_id;
    ->
    ->     IF ROW_COUNT() = 0 THEN
    ->         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Failed to deduct amount from sender';
    ->     END IF;
    ->
    ->     -- Add the transfer amount to the receiver
    ->     UPDATE accounts
    ->     SET balance = balance + transfer_amount
    ->     WHERE customer_id = receiver_id;
    ->
    ->     IF ROW_COUNT() = 0 THEN
    ->         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Failed to add amount to receiver';
    ->     END IF;
    ->
    ->     -- Commit transaction if no errors
    ->     COMMIT;
    ->
    ->     SELECT 'Transaction successful, changes committed' AS message;
    -> END$$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql>
```

Handles Error with Rollback.

```
mysql> SELECT * FROM accounts;
+----+-------------+---------+
| id | customer_id | balance |
+----+-------------+---------+
|  1 |           1 | 1000.00 |
|  2 |           2 | 2000.00 |
|  3 |           3 | 1500.00 |
+----+-------------+---------+
3 rows in set (0.00 sec)

mysql> CALL transaction_error_case(1, 99, 50.00); -- Receiver does not exist
+-----------------------------------+
| message                           |
+-----------------------------------+
| Transaction failed, rollback executed |
+-----------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> CALL transaction_error_case(1, 2, 5000.00); -- Amount exceeds sender's balance
+-----------------------------------+
| message                           |
+-----------------------------------+
| Transaction failed, rollback executed |
+-----------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM accounts;
+----+-------------+---------+
| id | customer_id | balance |
+----+-------------+---------+
|  1 |           1 | 1000.00 |
|  2 |           2 | 2000.00 |
|  3 |           3 | 1500.00 |
+----+-------------+---------+
3 rows in set (0.00 sec)
```

Commits if transaction successful.

```
mysql> CALL transaction_error_case(1, 2, 50.00);
+----------------------------------------+
| message                                |
+----------------------------------------+
| Transaction successful, changes committed |
+----------------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM accounts;
+----+-------------+---------+
| id | customer_id | balance |
+----+-------------+---------+
|  1 |           1 |  950.00 |
|  2 |           2 | 2050.00 |
|  3 |           3 | 1500.00 |
+----+-------------+---------+
3 rows in set (0.00 sec)
```