UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

ExperimentNo. 1.7

Student Name: Gaurav Kumar
UID: 22MCC20177
Branch: MCA–CCD
Section/Group: MCD-1/A
Semester: III
Date of Performance: 15th Oct 23
Subject Name: Business Analytics
Subject Code: 22CAH-703

1. **Aim/Overview of the practical:**
   a) Execute the JOIN in SQL by taking a table as an example. Explain the differences between an INNER JOIN and a LEFT JOIN in SQL.
   b) Describe the concept of a self-join in SQL. Provide an example scenario where a self-join would be useful.

2. **Code for practical: (a)**

Step 1 : To perform the join operation first create at least 2 tables.

```sql
create database BA_Practical;
use BA_Practical;
CREATE TABLE student (
    student_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    date_of_birth DATE,
    major VARCHAR(50)
);
CREATE TABLE student_marks (
    student_id INT,
    subject VARCHAR(50),
    marks INT
);
INSERT INTO student VALUES (1, 'John', 'Smith', '1995-02-15', 'Computer Science'),
(2, 'Alice', 'Johnson', '1998-07-20', 'Biology'),
(3, 'Michael', 'Brown', '1997-04-10', 'History'),
(4, 'Emily', 'Williams', '1999-12-05', 'Mathematics'),
(5, 'David', 'Jones', '1996-09-30', 'Chemistry');

INSERT INTO student_marks  VALUES (1, 'Math', 90), (1, 'Science', 85),
(2, 'Math', 78), (2, 'Science', 92), (4, 'Math', 86);
```

Step 2 : To perform join on 2 tables we can use **JOIN** keyword to join and **ON** keyword to specify relation on which the join is perform.

```
SELECT st.student_id, st.first_name, sm.subject, sm.marks
FROM student st JOIN student_marks sm
ON st.student_id = sm.student_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	first_name	subject	marks
1	John	Math	90
1	John	Science	85
2	Alice	Math	78
2	Alice	Science	92
4	Emily	Math	86

INNER JOIN: An INNER JOIN returns only the rows where there is a match in both tables. In this case, it will retrieve rows where students have corresponding records in the "student_marks" table. **Note: join** and **Inner join** both are same.

LEFT JOIN: A LEFT JOIN returns all rows from the left table (the "student" table) and the matched rows from the right table (the "student_marks" table). If there's no match in the right table for a row in the left table, the result will include NULL values for the right table's columns.

```
SELECT st.student_id, st.first_name, sm.subject, sm.marks
FROM student st LEFT JOIN student_marks sm
ON st.student_id = sm.student_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	first_name	subject	marks
1	John	Science	85
1	John	Math	90
2	Alice	Science	92
2	Alice	Math	78
3	Michael	NULL	NULL
4	Emily	Math	86
5	David	NULL	NULL

Code for practical: (b)

SELF JOIN: A self-join in SQL is a specific type of join operation where a table is joined with itself. In other words, it's a way to combine rows from a single table based on a related column within that same table. Self-joins are particularly useful when you have hierarchical or recursive data structures or when you want to establish relationships between rows in the same table.

Example Scenario: Let's consider an example scenario where a self-join would be useful: representing an organizational hierarchy. You have an "employees" table that includes data about employees and their managers. The table might have the following structure:

Field	Type	Null	Key	Default	Extra
employee_id	int	NO	PRI	NULL	
employee_name	varchar(50)	YES		NULL	
manager_id	int	YES		NULL	

To visualize the organizational hierarchy, you can use a self-join to connect employees with their managers. For example, to retrieve the names of employees and their managers, you can write a self-join query like this:

```
SELECT e.employee_name AS employee, m.employee_name AS manager
FROM employees e LEFT JOIN employees m
ON e.manager_id = m.employee_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

employee	manager
John	NULL
Alice	John
Michael	Alice
Emily	John
David	Emily

In this example, self-join helps establish relationships between employees and their managers, making it easier to represent and understand the organizational hierarchy within the company. Self-joins are particularly valuable for handling hierarchical or recursive data structures and for querying data with relationships between rows within the same table.