Joins BDSM

Aim: To understand and implement different types of SQL JOINs for combining data from multiple tables in a relational database.

Course Outcomes:

- **CO3:** Apply Relational Algebra and Relational Calculus Query to the database of an organization.

Theory:

In relational databases, data is often spread across multiple tables. JOIN operations allow us to combine related data from these tables based on a common column. This enables us to retrieve meaningful information that isn't isolated in a single table.

Types of JOINs:

- **INNER JOIN:** Returns rows only when there is a match in both tables based on the join condition.
- **LEFT (OUTER) JOIN:** Returns all rows from the left table, even if there is no match in the right table. Non-matching rows in the right table will have NULL values.
- **RIGHT (OUTER) JOIN:** Returns all rows from the right table, even if there is no match in the left table. Non-matching rows in the left table will have NULL values.
- **FULL (OUTER) JOIN:** Returns all rows from both tables. If there is no match on one side, the corresponding columns will contain NULL values.

Understanding JOINs:

JOINs are essential for querying related data in a relational database. They allow you to:

- Combine data from multiple tables into a single result set.
- Retrieve information that spans across different tables.
- Create complex queries to analyze and understand relationships between data.

Example:

Consider two tables, `Customers` and `Orders`. To retrieve customer names along with their order IDs, you would use a JOIN operation like this:

SQL

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
```

```
INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

This query joins the `Customers` and `Orders` tables based on the `CustomerID` column and retrieves the `CustomerName` from the `Customers` table and the `OrderID` from the `Orders` table for matching rows.

Learning Outcomes:

Upon completion of this experiment, students will be able to:

- Explain the purpose and benefits of using SQL JOINs.
- Differentiate between various types of JOINs (INNER, LEFT, RIGHT, FULL).
- Construct SQL queries using different JOINs to combine data from multiple tables.
- Analyze query results and understand the impact of different JOIN types.
- Apply JOINs to retrieve meaningful information from related tables in a database.

```
-- Creating the Customers table
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(255),
    ContactName VARCHAR(255),
    Country VARCHAR(255)
);

-- Creating the Orders table
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    OrderDate DATE,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

-- Inserting data into the Customers table
INSERT INTO Customers (CustomerID, CustomerName, ContactName, Country)
VALUES
    (1, 'Alfreds Futterkiste', 'Maria Anders', 'Germany'),
    (2, 'Ana Trujillo Emparedados y helados', 'Ana Trujillo', 'Mexico'),
    (3, 'Antonio Moreno Taquería', 'Antonio Moreno', 'Mexico'),
    (4, 'Around the Horn', 'Thomas Hardy', 'UK');

-- Inserting data into the Orders table
INSERT INTO Orders (OrderID, CustomerID, OrderDate) VALUES
    (10308, 2, '1996-09-18'),
```

```
    (10365, 3, '1996-11-27'),
    (10383, 4, '1996-12-16'),
    (10355, 4, '1996-11-15'),
    (10278, 1, '1996-08-12');

-- INNER JOIN: Retrieve all orders with customer information
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

-- LEFT JOIN: Retrieve all customers and their orders (if any)
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;

-- RIGHT JOIN: Retrieve all orders and their corresponding customers (if
any)
SELECT Orders.OrderID, Customers.CustomerName
FROM Customers
RIGHT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Orders.OrderID;
```