OUTER JOIN

In a relational DBMS, we follow the principles of normalization that allows us to minimize the large tables into small tables. By using a select statement in Joins, we can retrieve the big table back. Outer joins are of following three types.

1. Left outer join
2. Right outer join
3. Full outer join

Creating a database : Run the following command to create a database.

```
CREATE DATABASE testdb;
```

Using the database : Run the following command to use a database.

```
USE testdb;
```

Adding table to the database : Run the following command to add tables to a database.

```
CREATE TABLE Students (
    StudentID INT,
    LastName VARCHAR(255),
    FirstName VARCHAR(255),
    Address VARCHAR(255),
    City VARCHAR(255)
);
```

Inserting rows into database :

```
INSERT INTO Students (StudentID, LastName, FirstName,
Address, City)
VALUES (111, 'James', 'Johnson', 'USA', 'California');
```

Output of database :

Type the following command to get output.

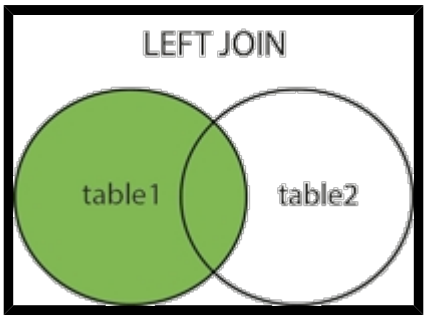
```
SELECT * FROM Students;
```

```
111|James|Johnson|USA|california

[Program exited with exit code 0]
```

Types of outer join

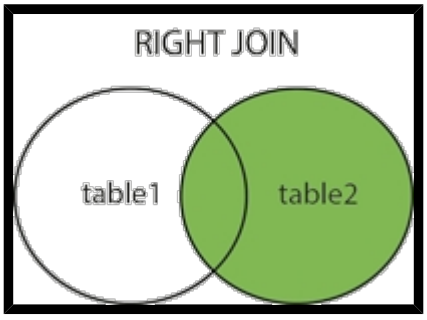
1. **Left Outer Join** : The left join operation returns all record from left table and matching records from the right table. On a matching element not found in right table, NULL is represented in that case.



Syntax

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

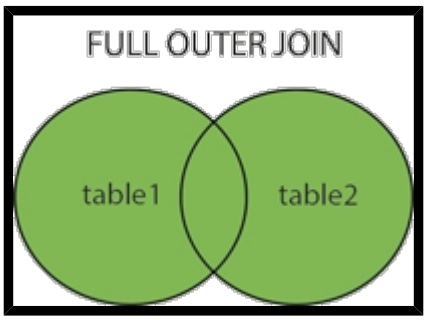
2. **Right Outer Join** : The right join operation returns all record from right table and matching records from the left table. On a matching element not found in left table, NULL is represented in that case.



Syntax

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

3. **Full Outer Join** : The full outer Join keyword returns all records when there is a match in left or right table records.



Syntax

```
SELECT column_name
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```

Example

Creating 1st Sample table students.

```
CREATE TABLE students (
    id INTEGER,
    name TEXT NOT NULL,
    gender TEXT NOT NULL
);
-- insert some values
INSERT INTO students VALUES (1, 'Ryan', 'M');
INSERT INTO students VALUES (2, 'Joanna', 'F');
INSERT INTO students VALUES (3, 'Moana', 'F');
```

Creating 2nd sample table college.

```
CREATE TABLE college (
    id INTEGER,
    classTeacher TEXT NOT NULL,
    Strength TEXT NOT NULL
);
-- insert some values
INSERT INTO college VALUES (1, 'Alpha', '50');
INSERT INTO college VALUES (2, 'Romeo', '60');
INSERT INTO college VALUES (3, 'Charlie', '55');
```

Performing outer join on above two tables.

```
SELECT College.classTeacher, Students.id
FROM College
FULL OUTER JOIN Students ON College.id = Students.id
```

```
ORDER BY College.classTeacher;
```

The above code will perform a full outer join on tables students and college and will return the output that matches the id of college with id of students. The output will be class Teacher from college table and id from students table. The table will be ordered by class Teacher from college table.

Class Teacher	Id
Alpha	1
Romeo	2
Charlie	3