

1. Table Structure

Students Table

```
sql  
  
CREATE TABLE Students (  
    StudentID INT PRIMARY KEY,  
    StudentName VARCHAR(100) NOT NULL,  
    DateOfBirth DATE,  
    TeacherID INT  
);
```

Teachers Table

```
CREATE TABLE Teachers (  
    TeacherID INT PRIMARY KEY,  
    TeacherName VARCHAR(100) NOT NULL,  
    Subject VARCHAR(100) NOT NULL  
);
```

2. Types of SQL Joins

a. INNER JOIN

The INNER JOIN returns only the rows where there is a match between the Students and Teachers tables based on the TeacherID.

The screenshot shows the phpMyAdmin interface for a database named 'school management'. The 'students' table is selected. The query executed is:

```
SELECT Students.StudentID, Students.StudentName, Teachers.TeacherName, Teachers.Subject FROM Students INNER JOIN Teachers ON Students.TeacherID = Teachers.TeacherID;
```

The query results show 10 rows, indicating that all students in the 'students' table have a corresponding entry in the 'teachers' table. The results are displayed in a table with the following columns: StudentID, StudentName, TeacherName, and Subject.

StudentID	StudentName	TeacherName	Subject
1	John	Mr. Adams	Math
2	Jane	Ms. Clark	English
3	Sam	Ms. Clark	English
4	Emily	Mrs. Green	History
5	David	Ms. Clark	English
6	Brown	Ms. Clark	English
7	White	Mrs. Green	History
8	Samown	Mr. Adams	Math
9	San	Ms. Clark	English
10	Emit	Mr. Adams	Math

Explanation:

- This query retrieves students who are associated with a teacher (i.e., TeacherID matches in both tables).
- If a student doesn't have a TeacherID, they will not be included in the result.

b. LEFT (OUTER) JOIN

The LEFT JOIN (also known as LEFT OUTER JOIN) returns all rows from the Students table and the matched rows from the Teachers table. If no match is found, NULL values are returned for the Teachers columns.

The screenshot shows the phpMyAdmin interface for a database named 'school management'. The 'students' table is selected, and a query is executed showing the results of a LEFT OUTER JOIN between the 'Students' and 'Teachers' tables. The query is: `SELECT Students.StudentID, Students.StudentName, Teachers.TeacherName, Teachers.Subject FROM Students LEFT OUTER JOIN Teachers ON Students.TeacherID = Teachers.TeacherID;`

The results table displays 10 rows of data. The columns are StudentID, StudentName, TeacherName, and Subject. The data is as follows:

StudentID	StudentName	TeacherName	Subject
1	John	Mr. Adams	Math
2	Jane	Ms. Clark	English
3	Sam	Ms. Clark	English
4	Emily	Mrs. Green	History
5	David	Ms. Clark	English
6	Brown	Ms. Clark	English
7	White	Mrs. Green	History
8	Samown	Mr. Adams	Math
9	San	Ms. Clark	English
10	Emit	Mr. Adams	Math

The interface also shows a message: 'Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.' and a status bar at the bottom indicating the system time as 11:55 AM on 11/10/2024.

Explanation:

- This query retrieves all students, regardless of whether they have a matching teacher.
- If a student does not have a TeacherID, NULL is returned for the teacher-related columns.

c. RIGHT (OUTER) JOIN

The RIGHT JOIN (also known as RIGHT OUTER JOIN) returns all rows from the Teachers table and the matched rows from the Students table. If no match is found, NULL values are returned for the Students columns.

The screenshot shows the phpMyAdmin interface for a database named 'school management'. The 'students' table is selected. The SQL query executed is:

```
SELECT Students.StudentID, Students.StudentName, Teachers.TeacherName, Teachers.Subject FROM Students RIGHT OUTER JOIN Teachers ON Students.TeacherID = Teachers.TeacherID;
```

The query results show 10 rows, displaying all teachers and their associated students. The columns are StudentID, StudentName, TeacherName, and Subject.

StudentID	StudentName	TeacherName	Subject
1	John	Mr. Adams	Math
2	Jane	Ms. Clark	English
3	Sam	Ms. Clark	English
4	Emily	Mrs. Green	History
5	David	Ms. Clark	English
6	Brown	Ms. Clark	English
7	White	Mrs. Green	History
8	Samown	Mr. Adams	Math
9	San	Ms. Clark	English
10	Emit	Mr. Adams	Math

Explanation:

- This query retrieves all teachers, regardless of whether they are associated with any students.
- If a teacher has no students (i.e., no matching TeacherID in the Students table), NULL is returned for the student-related columns.

d. FULL OUTER JOIN (Using UNION)

MySQL does not support FULL OUTER JOIN directly. However, you can simulate a FULL OUTER JOIN by combining a LEFT JOIN and a RIGHT JOIN using UNION.

The screenshot shows the phpMyAdmin interface for a database named 'school management'. The 'students' table is selected. The SQL query editor displays the following query:

```
SELECT Students.StudentID, Students.StudentName, Teachers.TeacherID, Teachers.TeacherName, Teachers.Subject FROM Students LEFT JOIN Teachers ON Students.TeacherID = Teachers.TeacherID UNION SELECT Students.StudentID, Students.StudentName, Teachers.TeacherID, Teachers.TeacherName, Teachers.Subject FROM Students RIGHT JOIN Teachers ON Students.TeacherID = Teachers.TeacherID;
```

The query results show 10 rows, combining data from both the 'students' and 'teachers' tables. The results are as follows:

StudentID	StudentName	TeacherID	TeacherName	Subject
1	John	101	Mr. Adams	Math
2	Jane	102	Ms. Clark	English
3	Sam	102	Ms. Clark	English
4	Emily	103	Mrs. Green	History
5	David	102	Ms. Clark	English
6	Brown	102	Ms. Clark	English
7	White	103	Mrs. Green	History
8	Samown	101	Mr. Adams	Math
9	San	102	Ms. Clark	English
10	Emit	101	Mr. Adams	Math

Explanation:

- **First Query (LEFT JOIN):** Returns all students and their associated teachers, or NULL for teachers if no match is found.
- **Second Query (RIGHT JOIN):** Returns all teachers and their associated students, or NULL for students if no match is found.
- The UNION combines the results of both queries and eliminates any duplicates, ensuring that all records from both tables are returned.
- This effectively simulates the behavior of a FULL OUTER JOIN in MySQL.

4. Testing the Queries

To test the queries:

1. Ensure you have populated the Students and Teachers tables with sample data.
2. Run each query individually to see the results for different types of joins.

```
INSERT INTO Students (StudentID, StudentName, DateOfBirth, TeacherID)
```

```
VALUES
```

```
(1, 'John', '2005-03-01', 101),
```

```
(2, 'Jane', '2004-07-10', 102),
```

```
(3, 'Sam', '2005-11-20', 102),
```

```
(4, 'Emily', '2005-05-15', 103),
```

```
(5, 'David', '2004-09-30', 102),
```

```
(6, 'Brown', '2005-11-25', 102),
```

```
(7, 'White', '2005-05-18', 103),
```

```
(8, 'Samown', '2005-11-22', 101),
```

```
(9, 'San', '2005-11-21', 102),
```

```
(10, 'Emit', '2005-05-13', 101);
```

```
INSERT INTO Teachers (TeacherID, TeacherName, Subject)
```

```
VALUES
```

```
(101, 'Mr. Adams', 'Math'),
```

```
(102, 'Ms. Clark', 'English'),
```

```
(103, 'Mrs. Green', 'History');
```