**1.Convert the xml code into the json code**

<bookstore>

<book>  
        <title>Harry Potter</title>  
        <author>J.K. Rowling</author>  
        <price>29.99</price>  
        <available>true</available>  
    </book>  
    <book>  
        <title>The Hobbit</title>  
        <author>J.R.R. Tolkien</author>  
        <price>19.99</price>  
        <available>false</available>  
    </book>  
</bookstore>

**Json:**

{  
  "bookstore": {  
    "book": [  
      {  
        "title": "Harry Potter",  
        "author": "J.K. Rowling",  
        "price": 29.99,  
        "available": true  
      },  
      {  
        "title": "The Hobbit",  
        "author": "J.R.R. Tolkien",  
        "price": 19.99,  
        "available": false  
      }  
    ]  
  }  
}

OUTPUT

Displaying the Data in a Table Format

Title Author Price Available

Harry Potter J.K. Rowling 29. Yes

The Hobbit J.R.R. Tolkien 19.99 No

**DISCRIPTION ON INNER JOIN,LEFT OUTER JOIN,RIGHT**

**OUTER JOIN,FULL OUTER JOIN**

1. **INNER JOIN**

An INNER JOIN returns records that have matching values in both tables.

Syntax:

SELECT columns

FROM table1

INNER JOIN table2

ON table1.common\_field = table2.common\_field;

**Example:** Given two tables, Employees and Departments

SELECT Employees.Name, Departments.DepartmentName

FROM Employees

INNER JOIN Departments

Employees.DepartmentID = Departments.DepartmentID;

This will only return employees who have a matching department.

**2. LEFT OUTER JOIN (or LEFT JOIN)**

A LEFT OUTER JOIN returns all records from the left table (table1), and the matched recordsfrom the right table (table2). The result is NULL from the right side, if there is no match.

Syntax:

SELECT columns

FROM table1

LEFT OUTER JOIN table2

ON table1.common\_field = table2.common\_field;

**Example:** If you want to find all employees and their department names, includingemployees who do not belong to any department:

SELECT Employees.Name, Departments.DepartmentName

FROM Employees

LEFT OUTER JOIN Departments

ON Employees.DepartmentID = Departments.DepartmentID;

This will return all employees, and NULL for the department name where there is no match.

**3. RIGHT OUTER JOIN (or RIGHT JOIN)**

A RIGHT OUTER JOIN returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

Syntax:

SELECT columns

FROM table1

RIGHT OUTER JOIN table2

ON table1.common\_field = table2.common\_field;

**Example:** If you want to find all departments and their employees, including departments that do not have any employees:

SELECT Employees.Name, Departments.DepartmentName

FROM Employees

RIGHT OUTER JOIN Department

ON Employees.DepartmentID = Departments.DepartmentID;

This will return all departments, and NULL for the employee name where there is no match.

**4. FULL OUTER JOIN**

A FULL OUTER JOIN returns all records when there is a match in either left (table1) or right (table2)table records. This means it returns all records from both tables, and fills in NULLs for missing matches on either side.

Syntax:

SELECT columns

FROM table1

FULL OUTER JOIN table2

ON table1.common\_field = table2.common\_field;

**Example:** If you want to find all employees and all departments, regardless of whether they have matching entries in the other table:

SELECT Employees.Name, Departments.DepartmentName

FROM Employees

FULL OUTER JOIN Departments

ON Employees.DepartmentID = Departments.DepartmentID;

This will return all employees and all departments, with NULL values where there is no match.

**Summary Table**

**Join Type Result**

INNER JOIN Only records with matching values in both tables.

LEFT OUTER JOIN All records from the left table, matched records from the

Right match if no NULL.

RIGHT OUTER JOIN All records from the right table, matched records from

the left table, NULL if no match.

FULL OUTER JOIN All records from both tables, NULL where there is no

Match.

These joins are fundamental in SQL for combining records from two or more tables based ona related column between them.

**Employee table**

employee\_id        first\_name       last\_name       department\_id

     1                                John                 Doe                     10

     2                                Jane                  Smith                  20

     3                                Mike                Johnson               30

    4                                Emily                  Davis                  10

**Department Table**

department\_id                       department\_name

  10                                                    HR

  20                                                   Sales

  30                                                    IT

  40                                                    Marketing

**1.INNER JOIN**

SELECT e.employee\_id, e.first\_name, e.last\_name, d.department\_name

FROM Employee e

INNER JOIN Department d ON e.department\_id = d.department\_id;

**2.LEFT OUTER JOIN**

SELECT e.employee\_id, e.first\_name, e.last\_name, d.department\_name

FROM Employee e

LEFT OUTER JOIN Department d ON e.department\_id = d.department\_id;

**3.RIGHT OUTER JOIN**

SELECT e.employee\_id, e.first\_name, e.last\_name, d.department\_name

FROM Employee e

RIGHT OUTER JOIN Department d ON e.department\_id = d.department\_id;

**4.FULL OUTER JOIN**

SELECT e.employee\_id, e.first\_name, e.last\_name, d.department\_name

FROM Employee e

FULL OUTER JOIN Department d ON e.department\_id = d.department\_id;

**Find duplicate records:**

**a) Based on first name:**

SELECT first\_name, COUNT(\*)

FROM Employee

GROUP BY first\_name

HAVING COUNT(\*) &gt; 1;

**b) Based on email:**

SELECT email, COUNT(\*)

FROM Employee

GROUP BY email

HAVING COUNT(\*) &gt; 1;

**c) Based on first name and last name:**

SELECT first\_name, last\_name, COUNT(\*)

FROM Employee

GROUP BY first\_name, last\_name

HAVING COUNT(\*) &gt; 1;

**d) Based on first name and email:**

SELECT first\_name, email, COUNT(\*)

FROM Employee

GROUP BY first\_name, email

HAVING COUNT(\*) &gt; 1;