3. Implementation of JOINS

Different types of joins can be used to combine data from two or more tables.

First create required tables with sample data

```
-- Create the Orders table
CREATE TABLE Orders
 ( OrderID NUMBER(5),
    CustomerID NUMBER(5),
    OrderName VARCHAR2(50));

-- Insert data into the Orders table
INSERT INTO Orders VALUES (1, 1, 'Order1');
 INSERT INTO Orders VALUES(2, 2, 'Order2');
 INSERT INTO Orders VALUES (3, 3, 'Order3');
INSERT INTO Orders VALUES(4, 5, 'Order4');

-- Create the Customers table
CREATE TABLE Customers
  ( CustomerID NUMBER(5),
    CustomerName VARCHAR2(5),
    Contact VARCHAR2(50));

-- Insert data into the Customers table
INSERT INTO Customers VALUES (1, 'Customer1', 'Contact1');
INSERT INTO Customers VALUES (2, 'Customer2', 'Contact2');
INSERT INTO Customers VALUES (3, 'Customer3', 'Contact3');
INSERT INTO Customers VALUES (4, 'Customer4', 'Contact4');
```

INNER JOIN: This type of join returns only the matching rows from both tables.

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

FROM Orders, Customers

WHERE Orders.CustomerID = Customers.CustomerID;
```

This query would return only the rows where the " CustomerID " value exists in both tables.

OUTER JOIN: This type of join returns all the rows from one table and matching rows from the other table. If there are no matching rows in the other table, NULL values are returned. Types of outer joins are: LEFT OUTER JOIN, RIGHT OUTER JOIN, and FULL OUTER JOIN.

LEFT OUTER JOIN: This returns all the rows from the left table and the matching rows from the right table, and NULL values for the non-matching rows from the right table.

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

FROM Orders, Customers

WHERE Orders.CustomerID = Customers.CustomerID(+);
```

This query would return all the rows from the Orders table and matching rows from the Customers table, with NULL values for non-matching rows.

RIGHT OUTER JOIN: This returns all the rows from the right table and the matching rows from the left table, and NULL values for the non-matching rows from the left table.

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

```
FROM Orders, Customers

WHERE Orders.CustomerID(+) = Customers.CustomerID;
```

This query would return all the rows from the Customers table and matching rows from the Orders table, with NULL values for non-matching rows.

FULL OUTER JOIN: This returns all the rows from both tables, with NULL values for the non-matching rows.

For example, to perform a FULL OUTER JOIN between the Students table and Marks table on the "Reg_No" column, the query would be:

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

FROM Orders

FULL OUTER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

This query would return all the rows from both tables, with NULL values for non-matching rows.

NATURAL JOIN: This type of join matches two tables based on the columns that have the same name and data type.

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

FROM Orders

NATURAL JOIN Customers;
```

This query would return only the columns that have the same name and data type in both tables, and only the rows where the "CustomerID" value exists in both tables.