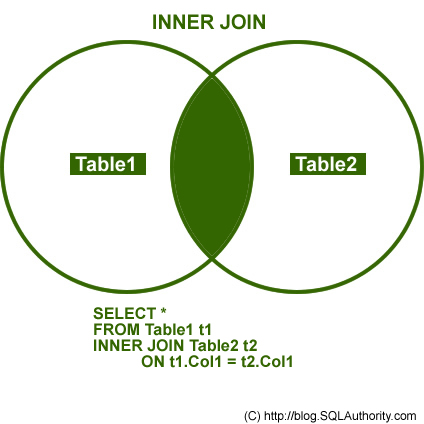
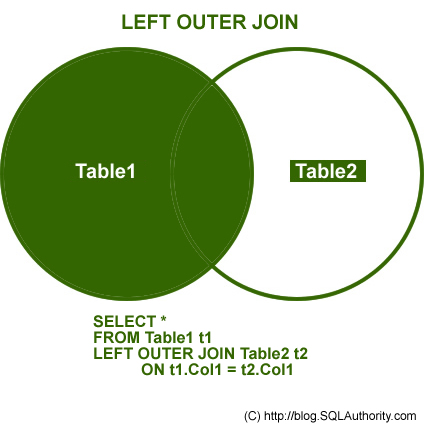
**INNER JOIN**

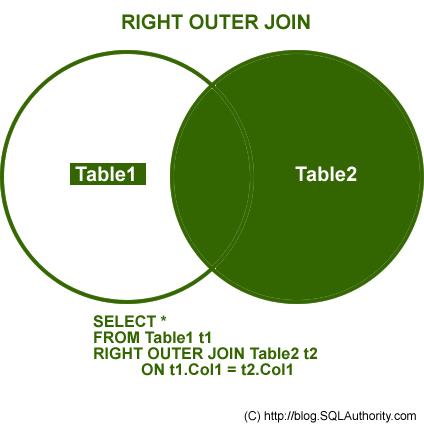
This join returns rows when there is at least one match in both the tables.  


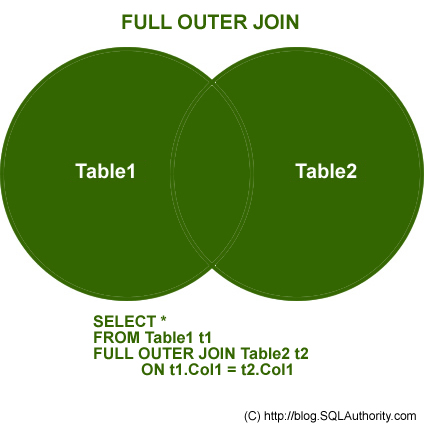
**OUTER JOIN**

There are three different Outer Join methods.

**LEFT OUTER JOIN**  
This join returns all the rows from the left table in conjunction with the matching rows from the right table. If there are no columns matching in the right table, it returns NULL values.  


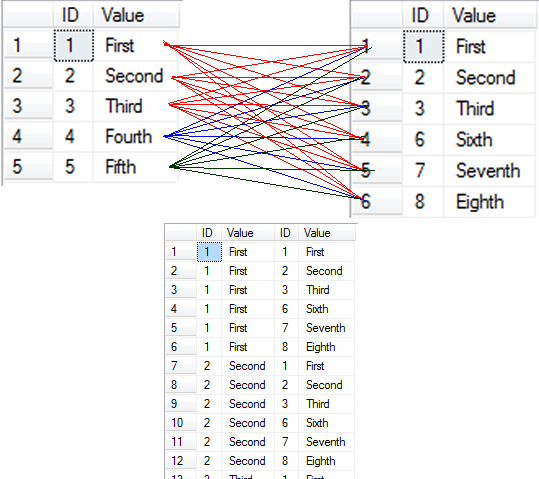
**RIGHT OUTER JOIN**  
This join returns all the rows from the right table in conjunction with the matching rows from the left table. If there are no columns matching in the left table, it returns NULL values.



**FULL OUTER JOIN**  
This join combines left outer join and right outer join. It returns row from either table when the conditions are met and returns null value when there is no match.  


CROSS JOIN

This join is a Cartesian join that does not necessitate any condition to join. The result set contains records that are multiplication of record number from both the tables.



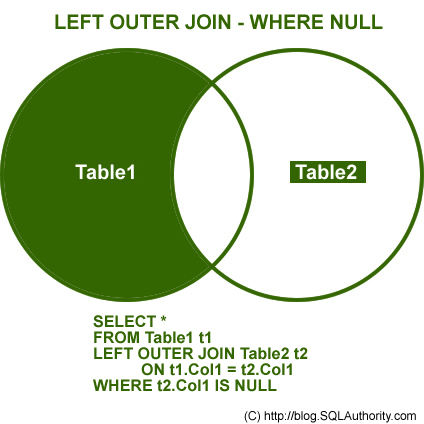
Additional Notes related to JOIN:

The following are three classic examples to display where Outer Join is useful. You will notice several instances where developers write query as given below.

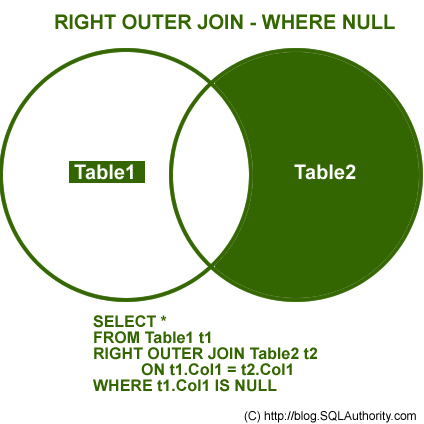
SELECT t1.\*  
FROM Table1 t1  
WHERE t1.ID NOT IN (SELECT t2.ID FROM Table2 t2)  
GO

The query demonstrated above can be easily replaced by Outer Join. Indeed, replacing it by Outer Join is the best practice. The query that gives same result as above is displayed here using Outer Join and WHERE clause in join.

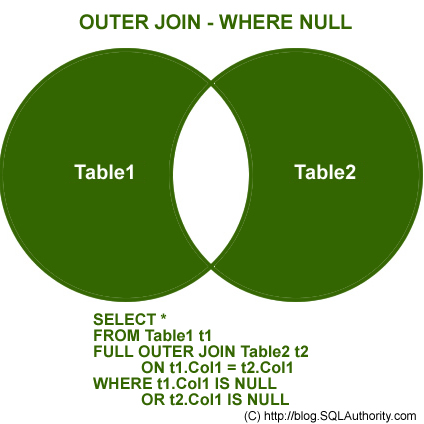
/\* LEFT JOIN - WHERE NULL \*/  
SELECT t1.\*,t2.\*  
FROM Table1 t1  
LEFT JOIN Table2 t2 ON t1.ID = t2.ID  
WHERE t2.ID IS NULL



The above example can also be created using Right Outer Join.



**NOT INNER JOIN**  
Remember, the term Not Inner Join does not exist in database terminology. However, when full Outer Join is used along with WHERE condition, as explained in the above two examples, it will give you exclusive result to Inner Join. This join will give all the results that were not present in Inner Join.



You can download the complete [**SQL Script**](http://www.pinaldave.com/sql-downloads/sql-downloads/script-downloads/sql-server-introduction-to-joins-basic-of-joins/) here, but for the sake of complicity I am including the same script here.

USE AdventureWorks  
GO  
CREATE TABLE table1  
(ID INT, Value VARCHAR(10))  
INSERT INTO Table1 (ID, Value)  
SELECT 1,'First'  
UNION ALL  
SELECT 2,'Second'  
UNION ALL  
SELECT 3,'Third'  
UNION ALL  
SELECT 4,'Fourth'  
UNION ALL  
SELECT 5,'Fifth'  
GO  
CREATE TABLE table2  
(ID INT, Value VARCHAR(10))  
INSERT INTO Table2 (ID, Value)  
SELECT 1,'First'  
UNION ALL  
SELECT 2,'Second'  
UNION ALL  
SELECT 3,'Third'  
UNION ALL  
SELECT 6,'Sixth'  
UNION ALL  
SELECT 7,'Seventh'  
UNION ALL  
SELECT 8,'Eighth'  
GO  
SELECT \*  
FROM Table1  
SELECT \*  
FROM Table2  
GO  
USE AdventureWorks  
GO  
/\* INNER JOIN \*/  
SELECT t1.\*,t2.\*  
FROM Table1 t1  
INNER JOIN Table2 t2 ON t1.ID = t2.ID  
GO  
/\* LEFT JOIN \*/  
SELECT t1.\*,t2.\*  
FROM Table1 t1  
LEFT JOIN Table2 t2 ON t1.ID = t2.ID  
GO  
/\* RIGHT JOIN \*/  
SELECT t1.\*,t2.\*  
FROM Table1 t1  
RIGHT JOIN Table2 t2 ON t1.ID = t2.ID  
GO  
/\* OUTER JOIN \*/  
SELECT t1.\*,t2.\*  
FROM Table1 t1  
FULL OUTER JOIN Table2 t2 ON t1.ID = t2.ID  
GO  
/\* LEFT JOIN - WHERE NULL \*/  
SELECT t1.\*,t2.\*  
FROM Table1 t1  
LEFT JOIN Table2 t2 ON t1.ID = t2.ID  
WHERE t2.ID IS NULL  
GO  
/\* RIGHT JOIN - WHERE NULL \*/  
SELECT t1.\*,t2.\*  
FROM Table1 t1  
RIGHT JOIN Table2 t2 ON t1.ID = t2.ID  
WHERE t1.ID IS NULL  
GO  
/\* OUTER JOIN - WHERE NULL \*/  
SELECT t1.\*,t2.\*  
FROM Table1 t1  
FULL OUTER JOIN Table2 t2 ON t1.ID = t2.ID  
WHERE t1.ID IS NULL OR t2.ID IS NULL  
GO  
/\* CROSS JOIN \*/  
SELECT t1.\*,t2.\*  
FROM Table1 t1  
CROSS JOIN Table2 t2  
GO  
DROP TABLE table1  
DROP TABLE table2  
GO

USE master

GO

IF EXISTS(SELECT name FROM sys.databases WHERE name = N'JoinExample')

DROP DATABASE JoinExample;

GO

CREATE DATABASE JoinExample;

GO

USE JoinExample

GO

IF OBJECT\_ID('Table1', 'U') IS NOT NULL

DROP TABLE Table1

GO

CREATE TABLE Table1

(ID INT, Value VARCHAR(10))

INSERT INTO Table1 (ID, Value)

SELECT 1,'First'

UNION ALL

SELECT 2,'Second'

UNION ALL

SELECT 3,'Third'

UNION ALL

SELECT 4,'Fourth'

UNION ALL

SELECT 5,'Fifth'

GO

USE JoinExample

GO

IF OBJECT\_ID('Table2', 'U') IS NOT NULL

DROP TABLE Table2

GO

CREATE TABLE Table2

(ID INT, Value VARCHAR(10))

INSERT INTO Table2 (ID, Value)

SELECT 1,'I'

UNION ALL

SELECT 2,'II'

UNION ALL

SELECT 3,'III'

UNION ALL

SELECT 6,'VI'

UNION ALL

SELECT 7,'VII'

UNION ALL

SELECT 8,'VIII'

GO

SELECT \* FROM Table1;

SELECT \* FROM Table2;

GO

/\* 1. INNER JOIN \*/

SELECT t1.\*,t2.\*

FROM Table1 t1

INNER JOIN Table2 t2 ON t1.ID = t2.ID

GO

/\* 2. LEFT JOIN \*/

SELECT t1.\*,t2.\*

FROM Table1 t1

LEFT JOIN Table2 t2 ON t1.ID = t2.ID

GO

/\* 3. RIGHT JOIN \*/

SELECT t1.\*,t2.\*

FROM Table1 t1

RIGHT JOIN Table2 t2 ON t1.ID = t2.ID

GO

/\* 4. OUTER JOIN \*/

SELECT t1.\*,t2.\*

FROM Table1 t1

FULL OUTER JOIN Table2 t2 ON t1.ID = t2.ID

GO

/\* 5. LEFT JOIN - WHERE NULL \*/

SELECT t1.\*,t2.\*

FROM Table1 t1

LEFT JOIN Table2 t2 ON t1.ID = t2.ID

WHERE t2.ID IS NULL

GO

/\* 6. RIGHT JOIN - WHERE NULL \*/

SELECT t1.\*,t2.\*

FROM Table1 t1

RIGHT JOIN Table2 t2 ON t1.ID = t2.ID

WHERE t1.ID IS NULL

GO

/\* 7. OUTER JOIN - WHERE NULL \*/

SELECT t1.\*,t2.\*

FROM Table1 t1

FULL OUTER JOIN Table2 t2 ON t1.ID = t2.ID

WHERE t1.ID IS NULL OR t2.ID IS NULL

GO

/\* 8. CROSS JOIN \*/

SELECT t1.\*,t2.\*

FROM Table1 t1

CROSS JOIN Table2 t2

GO

**Self Join**

If there are three kinds of joins, i.e. - Inner Join, Outer Join and Cross Join; what type of join is Self Join? In fact, it can be classified under any type of join.

Let us create a new example today, where we will see how Self Join can be implemented as an Inner Join as well as Outer Join.

Let us first create the same table for an employee. One of the columns in the same table contains the ID of manger, who is also an employee for the same company. This way, all the employees and their managers are present in the same table. If we want to find the manager of a particular employee, we need use self join.

USE [master];

Go

--IF EXISTS(SELECT name FROM sys.databases WHERE name = N'Testdb')

--DROP DATABASE [Testdb];

--GO

-- OR

IF DB\_ID (N'Testdb') IS NOT NULL

DROP DATABASE [Testdb];

GO

CREATE DATABASE [Testdb];

GO

USE [Testdb];

GO

--IF EXISTS(SELECT name FROM sys.tables WHERE name = N'Test')

--DROP TABLE [Employee];

--GO

-- OR

IF OBJECT\_ID (N'Employee', N'U') IS NOT NULL

DROP TABLE [Employee];

GO

-- Create a Table

CREATE TABLE Employee

(

EmployeeID INT PRIMARY KEY,

Name NVARCHAR(50),

ManagerID INT

)

GO

-- Insert Sample Data

INSERT INTO Employee

SELECT 1, 'Mike', 3

UNION ALL

SELECT 2, 'David', 3

UNION ALL

SELECT 3, 'Roger', NULL

UNION ALL

SELECT 4, 'Marry',2

UNION ALL

SELECT 5, 'Joseph',2;

GO

/\* Inner and Outer Join as Self Join \*/

IF OBJECT\_ID('Employee', 'U') IS NOT NULL

DROP TABLE Employee

GO

-- Create a Table

CREATE TABLE Employee

(

EmployeeID INT PRIMARY KEY,

Name NVARCHAR(50),

ManagerID INT

)

GO

-- Insert Sample Data

INSERT INTO Employee

SELECT 1, 'Mike', 3

UNION ALL

SELECT 2, 'David', 3

UNION ALL

SELECT 3, 'Roger', NULL

UNION ALL

SELECT 4, 'Marry',2

UNION ALL

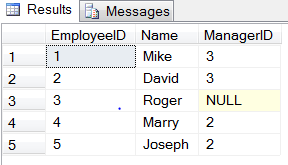
SELECT 5, 'Joseph',2;

GO

-- Check the data

SELECT \* FROM Employee;

GO



-- Inner Join as Self Join

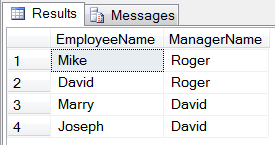
SELECT E1.Name AS EmployeeName, E2.Name AS ManagerName

FROM Employee E1

INNER JOIN Employee E2

ON E1.ManagerID = E2.EmployeeID;

GO



-- Outer Join as Self Join

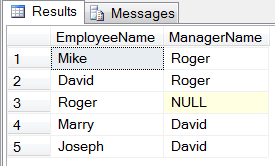
SELECT E1.Name AS EmployeeName, E2.name AS ManagerName

FROM Employee E1

LEFT JOIN Employee E2

ON E1.ManagerID = E2.EmployeeID;

GO



-- Outer Join as Self Join

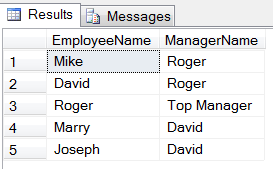
SELECT E1.Name AS EmployeeName, ISNULL(E2.name, 'Top Manager') AS ManagerName

FROM Employee E1

LEFT JOIN Employee E2

ON E1.ManagerID = E2.EmployeeID;

GO



Once we convert Inner Join to Outer Join, we can see the Top Manager as well. Here we have seen how Self Join can behave as an inner join as well as an outer join.

<http://blog.sqlauthority.com/2009/04/13/sql-server-introduction-to-joins-basic-of-joins/>

<http://blog.sqlauthority.com/2010/07/08/sql-server-the-self-join-inner-join-and-outer-join/>