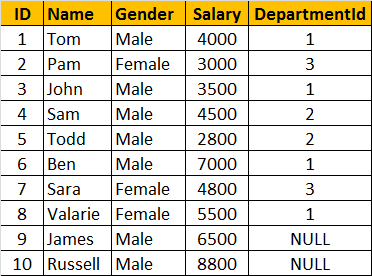
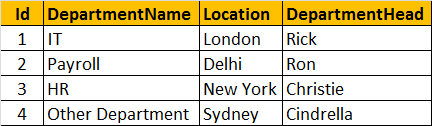
**,,Joins in SQL server** are used to query (retrieve) data from 2 or more related tables. In general tables are related to each other using foreign key constraints.

**n SQL server, there are different types of JOINS.**  
1. CROSS JOIN  
2. INNER JOIN   
3. OUTER JOIN   
  
**Outer Joins are again divided into 3 types**  
1. Left Join or Left Outer Join  
2. Right Join or Right Outer Join`  
3. Full Join or Full Outer Join  
  
**Now let's understand all the JOIN types, with examples and the differences between them.**  
**Employee Table (tblEmployee)**  
  
  
**Departments Table (tblDepartment)**  
  
  
**SQL Script to create tblEmployee and tblDepartment tables**

Create table tblDepartment

(

     ID int primary key,

     DepartmentName nvarchar(50),

     Location nvarchar(50),

     DepartmentHead nvarchar(50)

)

Go

Insert into tblDepartment values (1, 'IT', 'London', 'Rick')

Insert into tblDepartment values (2, 'Payroll', 'Delhi', 'Ron')

Insert into tblDepartment values (3, 'HR', 'New York', 'Christie')

Insert into tblDepartment values (4, 'Other Department', 'Sydney', 'Cindrella')

Go

Create table tblEmployee

(

     ID int primary key,

     Name nvarchar(50),

     Gender nvarchar(50),

     Salary int,

     DepartmentId int foreign key references tblDepartment(Id)

)

Go

Insert into tblEmployee values (1, 'Tom', 'Male', 4000, 1)

Insert into tblEmployee values (2, 'Pam', 'Female', 3000, 3)

Insert into tblEmployee values (3, 'John', 'Male', 3500, 1)

Insert into tblEmployee values (4, 'Sam', 'Male', 4500, 2)

Insert into tblEmployee values (5, 'Todd', 'Male', 2800, 2)

Insert into tblEmployee values (6, 'Ben', 'Male', 7000, 1)

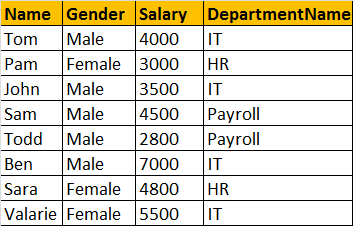
Insert into tblEmployee values (7, 'Sara', 'Female', 4800, 3)

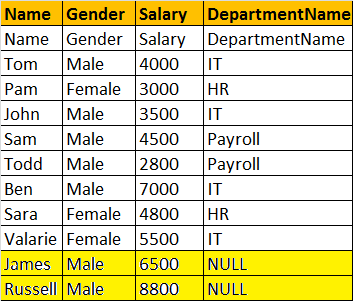
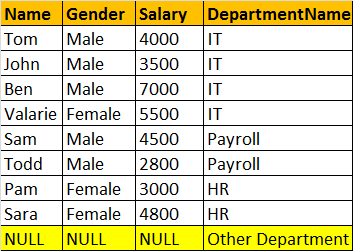
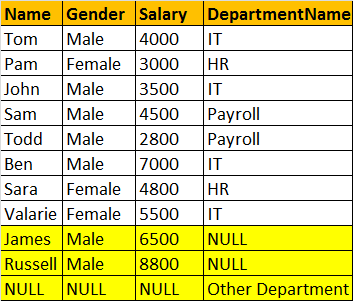
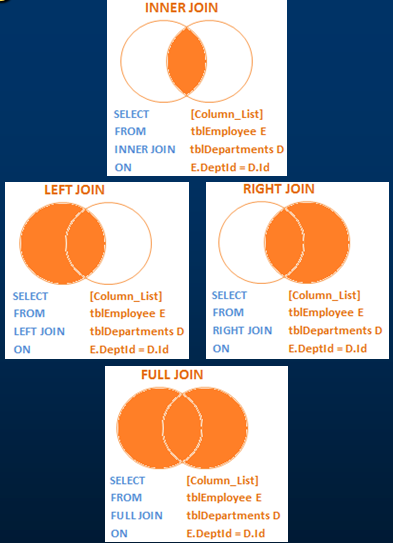
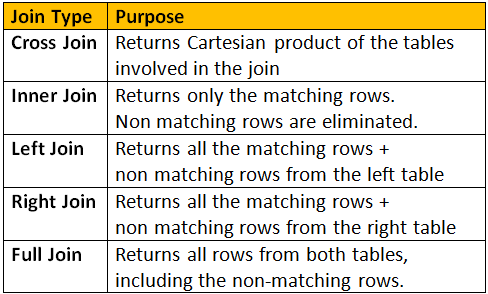
Insert into tblEmployee values (8, 'Valarie', 'Female', 5500, 1)

Insert into tblEmployee values (9, 'James', 'Male', 6500, NULL)

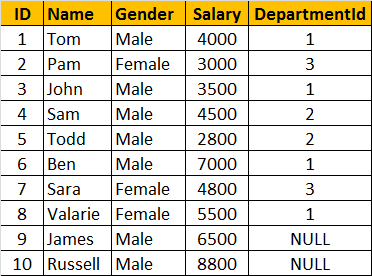
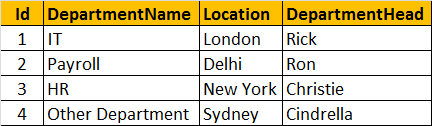
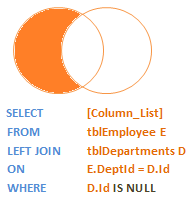
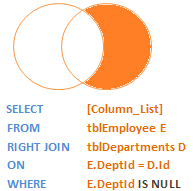
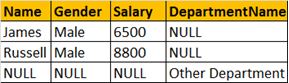
Insert into tblEmployee values (10, 'Russell', 'Male', 8800, NULL)

Go

**General Formula for Joins**  
SELECT      ColumnList  
FROM           LeftTableName  
JOIN\_TYPE  RightTableName  
ON                 JoinCondition  
  
**CROSS JOIN**  
CROSS JOIN, produces the cartesian product of the 2 tables involved in the join. For example, in the Employees table we have 10 rows and in the Departments table we have 4 rows. So, a cross join between these 2 tables produces 40 rows. Cross Join shouldn't have ON clause.   
  
**CROSS JOIN Query:**  
SELECT Name, Gender, Salary, DepartmentName  
FROM tblEmployee  
CROSS JOIN tblDepartment  
  
**JOIN or INNER JOIN**  
Write a query, to retrieve Name, Gender, Salary and DepartmentName from Employees and Departments table. The output of the query should be as shown below.  
  
  
SELECT Name, Gender, Salary, DepartmentName  
FROM tblEmployee  
INNER JOIN tblDepartment  
ON tblEmployee.DepartmentId = tblDepartment.Id  
  
**OR**  
  
SELECT Name, Gender, Salary, DepartmentName  
FROM tblEmployee  
JOIN tblDepartment  
ON tblEmployee.DepartmentId = tblDepartment.Id  
  
**Note:** JOIN or INNER JOIN means the same. It's always better to use INNER JOIN, as this explicitly specifies your intention.  
  
If you look at the output, we got only 8 rows, but in the Employees table, we have 10 rows. We didn't get JAMES and RUSSELL records. This is because the DEPARTMENTID, in Employees table is NULL for these two employees and doesn't match with ID column in Departments table.

So, in summary, INNER JOIN, returns only the matching rows between both the tables. Non matching rows are eliminated.  
  
**LEFT JOIN or LEFT OUTER JOIN**  
Now, let's say, I want all the rows from the Employees table, including JAMES and RUSSELL records. I want the output, as shown below.  
  
  
SELECT Name, Gender, Salary, DepartmentName  
FROM tblEmployee  
LEFT OUTER JOIN tblDepartment  
ON tblEmployee.DepartmentId = tblDepartment.Id  
  
**OR**  
  
SELECT Name, Gender, Salary, DepartmentName  
FROM tblEmployee  
LEFT JOIN tblDepartment  
ON tblEmployee.DepartmentId = tblDepartment.Id  
  
**Note:** You can use, LEFT JOIN or LEFT OUTER JOIN. OUTER keyowrd is optional  
  
**LEFT JOIN**, returns all the matching rows + non matching rows from the left table. In reality, INNER JOIN and LEFT JOIN are extensively used.  
  
**RIGHT JOIN or RIGHT OUTER JOIN**  
I want, all the rows from the right table. The query output should be, as shown below.  
  
  
SELECT Name, Gender, Salary, DepartmentName  
FROM tblEmployee  
RIGHT OUTER JOIN tblDepartment  
ON tblEmployee.DepartmentId = tblDepartment.Id  
  
**OR**  
  
SELECT Name, Gender, Salary, DepartmentName  
FROM tblEmployee  
RIGHT JOIN tblDepartment  
ON tblEmployee.DepartmentId = tblDepartment.Id  
  
**Note:** You can use, RIGHT JOIN or RIGHT OUTER JOIN. OUTER keyowrd is optional  
  
**RIGHT JOIN**, returns all the matching rows + non matching rows from the right table.  
  
**FULL JOIN or FULL OUTER JOIN**  
I want all the rows from both the tables involved in the join. The query output should be, as shown below.  
  
  
  
SELECT Name, Gender, Salary, DepartmentName  
FROM tblEmployee  
FULL OUTER JOIN tblDepartment  
ON tblEmployee.DepartmentId = tblDepartment.Id  
  
OR  
  
SELECT Name, Gender, Salary, DepartmentName  
FROM tblEmployee  
FULL JOIN tblDepartment  
ON tblEmployee.DepartmentId = tblDepartment.Id  
  
**Note:** You can use, FULLJOIN or FULL OUTER JOIN. OUTER keyowrd is optional  
  
**FULL JOIN**, returns all rows from both the left and right tables, including the non matching rows.  
  
**Joins Summary**  
  
  


Left Join 🡺 A

**Considers Employees (tblEmployee) and Departments (tblDepartment) tables**  
  
**Employee Table (tblEmployee)**  
  
  
**Departments Table (tblDepartment)**  
  
  
**How to retrieve only the non matching rows from the left table. The output should be as shown below:**  
  
  
**Query:**  
SELECT       Name, Gender, Salary, DepartmentName  
FROM           tblEmployee E  
LEFT JOIN   tblDepartment D  
ON                 E.DepartmentId = D.Id  
WHERE        D.Id IS NULL  
  
  
  
**How to retrieve only the non matching rows from the right table**  
  
  
**Query:**  
SELECT         Name, Gender, Salary, DepartmentName  
FROM             tblEmployee E  
RIGHT JOIN    tblDepartment D  
ON                   E.DepartmentId = D.Id  
WHERE          E.DepartmentId IS NULL  
  
  
  
**How to retrieve only the non matching rows from both the left and right table. Matching rows should be eliminated.**  
  
  
**Query:**  
SELECT         Name, Gender, Salary, DepartmentName  
FROM              tblEmployee E  
FULL JOIN      tblDepartment D  
ON                   E.DepartmentId = D.Id  
WHERE          E.DepartmentId IS NULL  
OR                   D.Id IS NULL  
  


**Self Join**

A self join is used to join a table to itself. This can be useful for querying hierarchical data or comparing rows within the same table. Let’s consider an example with a table named Employees that has the following columns: EmployeeID, Name, and ManagerID.

drop table Employees

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

Name NVARCHAR(50),

ManagerID INT

);

INSERT INTO Employees (EmployeeID, Name, ManagerID) VALUES

(1, 'John Doe', NULL),

(2, 'Jane Smith', 1),

(3, 'Jim Brown', 1),

(4, 'Jake White', 2),

(5, 'Jill Green', 2);

SELECT

e1.Name AS Employee,

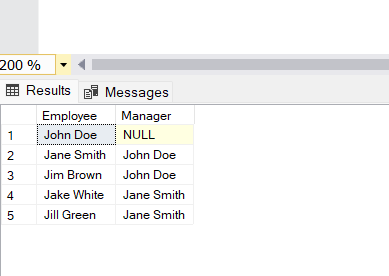
e2.Name AS Manager

FROM

Employees e1

LEFT JOIN

Employees e2 ON e1.ManagerID = e2.EmployeeID;



Group by Example

