In this video we will discuss **cross apply and outer apply in sql server** with examples.  
  
   
  
We will use the following 2 tables for examples in this demo

|  |  |
| --- | --- |
| department table | employee table |

SQL Script to create the tables and populate with test data

Create table Department

(

    Id int primary key,

    DepartmentName nvarchar(50)

)

Go

Insert into Department values (1, 'IT')

Insert into Department values (2, 'HR')

Insert into Department values (3, 'Payroll')

Insert into Department values (4, 'Administration')

Insert into Department values (5, 'Sales')

Go

Create table Employee

(

    Id int primary key,

    Name nvarchar(50),

    Gender nvarchar(10),

    Salary int,

    DepartmentId int foreign key references Department(Id)

)

Go

Insert into Employee values (1, 'Mark', 'Male', 50000, 1)

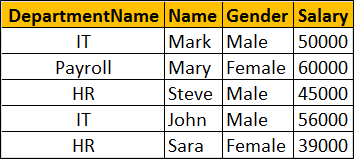
Insert into Employee values (2, 'Mary', 'Female', 60000, 3)

Insert into Employee values (3, 'Steve', 'Male', 45000, 2)

Insert into Employee values (4, 'John', 'Male', 56000, 1)

Insert into Employee values (5, 'Sara', 'Female', 39000, 2)

Go

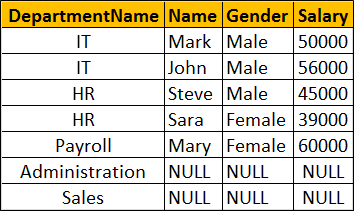
We want to retrieve all the matching rows between **Department**and **Employee**tables.   
   
  
This can be very easily achieved using an Inner Join as shown below.

Select D.DepartmentName, E.Name, E.Gender, E.Salary

from Department D

Inner Join Employee E

On D.Id = E.DepartmentId

Now if we want to retrieve all the matching rows between **Department**and **Employee**tables + the non-matching rows from the LEFT table (**Department**)   
   
  
This can be very easily achieved using a Left Join as shown below.

Select D.DepartmentName, E.Name, E.Gender, E.Salary

from Department D

Left Join Employee E

On D.Id = E.DepartmentId

Now let's assume we do not have access to the Employee table. Instead we have access to the following Table Valued function, that returns all employees belonging to a department by Department Id. 

Create function fn\_GetEmployeesByDepartmentId(@DepartmentId int)

Returns Table

as

Return

(

    Select Id, Name, Gender, Salary, DepartmentId

    from Employee where DepartmentId = @DepartmentId

)

Go

The following query returns the employees of the department with Id =1.

Select \* from fn\_GetEmployeesByDepartmentId(1)

Now if you try to perform an Inner or Left join between **Department**table and **fn\_GetEmployeesByDepartmentId**() function you will get an error. 

Select D.DepartmentName, E.Name, E.Gender, E.Salary

from Department D

Inner Join fn\_GetEmployeesByDepartmentId(D.Id) E

On D.Id = E.DepartmentId

If you execute the above query you will get the following error  
Msg 4104, Level 16, State 1, Line 3  
The multi-part identifier "D.Id" could not be bound.   
  
This is where we use **Cross Apply** and **Outer Apply** operators. **Cross Apply** is semantically equivalent to **Inner Join**and **Outer Apply**is semantically equivalent to **Left Outer Join**.

Just like Inner Join, Cross Apply retrieves only the matching rows from the Department table and fn\_GetEmployeesByDepartmentId() table valued function. 

Select D.DepartmentName, E.Name, E.Gender, E.Salary

from Department D

Cross Apply fn\_GetEmployeesByDepartmentId(D.Id) E

Just like Left Outer Join, Outer Apply retrieves all matching rows from the Department table and fn\_GetEmployeesByDepartmentId() table valued function + non-matching rows from the left table (Department) 

Select D.DepartmentName, E.Name, E.Gender, E.Salary

from Department D

Outer Apply fn\_GetEmployeesByDepartmentId(D.Id) E

**How does Cross Apply and Outer Apply work**

* The APPLY operator introduced in SQL Server 2005, is used to join a table to a table-valued function.
* The Table Valued Function on the right hand side of the APPLY operator gets called for each row from the left (also called outer table) table.
* Cross Apply returns only matching rows (semantically equivalent to Inner Join)
* Outer Apply returns matching + non-matching rows (semantically equivalent to Left Outer Join). The unmatched columns of the table valued function will be set to NULL.