Contents

[1. Prerequisite 1](#_Toc373171087)

[2. JOIN 2](#_Toc373171088)

[INNER JOIN 2](#_Toc373171089)

[LEFT JOIN 3](#_Toc373171090)

[RIGHT JOIN 3](#_Toc373171091)

[FULL JOIN 4](#_Toc373171092)

[CROSS JOIN 4](#_Toc373171093)

[SELF JOIN 5](#_Toc373171094)

[LEFT EXCLUDING JOIN 6](#_Toc373171095)

[RIGHT EXCLUDING JOIN 6](#_Toc373171096)

[OUTER JOIN EXCLUDING JOIN 6](#_Toc373171097)

[JOINING 3 OR MORE TABLES 6](#_Toc373171098)

[3. Subquery 7](#_Toc373171099)

[Subqueries with Aliases 7](#_Toc373171100)

[Subqueries with IN 7](#_Toc373171101)

[Subqueries with NOT IN 7](#_Toc373171102)

[Subqueries in UPDATE 8](#_Toc373171103)

[Subqueries with EXISTS 8](#_Toc373171104)

[Subqueries with NOT EXISTS 8](#_Toc373171105)

[4. Common Table Expression 9](#_Toc373171106)

[Recursive CTE 9](#_Toc373171107)

[5. RANK function 10](#_Toc373171108)

[ROW\_NUMBER 10](#_Toc373171109)

[RANK 10](#_Toc373171110)

[DENSE\_RANK 11](#_Toc373171111)

[NTITLE 11](#_Toc373171112)

[6. SQL Code Practice 11](#_Toc373171113)

# Prerequisite

Demos for this day use

1) AdventureWorks database from <http://msftdbprodsamples.codeplex.com/releases/view/93587>

2) In Fsoft\_Training database which is created in Day2, run below script



This script create 2 tables: Table1 and Table2

Table1:

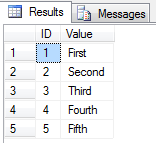
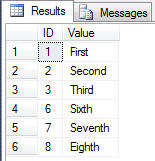


Table2:



# JOIN

## INNER JOIN

Example 1: Copy and **Execute** below statement.

USE AdventureWorks;

GO

SELECT

e.EmployeeID

, c.FirstName

, c.LastName

, e.NationalIDNumber

, e.Title

, e.BirthDate

, e.MaritalStatus

FROM HumanResources.Employee AS e

JOIN Person.Contact AS c

ON e.ContactID = c.ContactID

Example 2:

USE Fsoft\_Training

GO

SELECT t1.ID t1ID, t1.Value t1Value,t2.ID t2ID, t2.Value t2Value

FROM dbo.Table1 t1

INNER JOIN dbo.Table2 t2

ON t1.ID = t2.ID

GO

## LEFT JOIN

Example 1: Copy and run this statement. This **LEFT OUTER JOIN** includes all rows in the *Product* table in the results, whether or not there is a match on the ProductID column in the ProductReview table. Notice that in the results where there is no matching product review ID for a product, the row contains a null value in the *ProductReviewID* column

USE AdventureWorks;

GO

SELECT p.Name, pr.ProductReviewID

FROM Production.Product p

LEFT OUTER JOIN Production.ProductReview pr

ON p.ProductID = pr.ProductID

Example 2:

USE Fsoft\_Training

GO

SELECT t1.ID t1ID, t1.Value t1Value,t2.ID t2ID, t2.Value t2Value

FROM dbo.Table1 t1

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

GO

## RIGHT JOIN

Example 1: Include all sales persons in the results, regardless of whether they are assigned a territory

USE AdventureWorks;

GO

SELECT st.Name AS Territory, sp.SalesPersonID

FROM Sales.SalesTerritory st

RIGHT OUTER JOIN Sales.SalesPerson sp

ON st.TerritoryID = sp.TerritoryID ;

Example 2:

USE Fsoft\_Training

GO

SELECT t1.ID t1ID, t1.Value t1Value,t2.ID t2ID, t2.Value t2Value

FROM dbo.Table1 t1

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

GO

## FULL JOIN

Example 1: Returns all rows from both tables are to be included in the results, regardless of whether there is matching data in the tables

USE AdventureWorks;

GO

SELECT p.Name, sod.SalesOrderID

FROM Production.Product p

FULL OUTER JOIN Sales.SalesOrderDetail sod

ON p.ProductID = sod.ProductID

Example 2:

USE Fsoft\_Training

GO

SELECT t1.ID t1ID, t1.Value t1Value,t2.ID t2ID, t2.Value t2Value

FROM Table1 t1

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

## CROSS JOIN

Example 1:

USE AdventureWorks;

GO

SELECT p.SalesPersonID, t.Name AS Territory

FROM Sales.SalesPerson p

CROSS JOIN Sales.SalesTerritory t

ORDER BY p.SalesPersonID;

Example 2:

USE Fsoft\_Training

GO

SELECT t1.ID t1ID, t1.Value t1Value,t2.ID t2ID, t2.Value t2Value

FROM dbo.Table1 t1

CROSS JOIN dbo.Table2 t2

## SELF JOIN

Example 1: The following example performs a self-join of the HumanResources (use INNER JOIN) .Employee table to produce a list of all the managers and the employees that report to them.

USE AdventureWorks;

GO

SELECT MgrTable.LoginID AS ManagerName,EmplTable.ManagerID,

EmplTable.LoginID, EmplTable.EmployeeID

FROM HumanResources.Employee AS EmplTable

JOIN HumanResources.Employee AS MgrTable

ON EmplTable.ManagerID = MgrTable.EmployeeID

ORDER BY MgrTable.LoginID, EmplTable.LoginID

Example 2: The following example performs a self-join of the HumanResources (use LEFT JOIN).Employee table to produce a list of all the managers and the employees that report to them, it also specific who is top manager (who have ManagerID is null)

USE AdventureWorks;

GO

SELECT e1.LoginID EmployeeName, ISNULL(e2.LoginID, 'Top Manager') AS ManagerName

FROM HumanResources.Employee e1

LEFT JOIN HumanResources.Employee e2

ON e1.ManagerID = e2.EmployeeID

GO

## LEFT EXCLUDING JOIN

USE Fsoft\_Training

GO

SELECT t1.ID t1ID, t1.Value t1Value,t2.ID t2ID, t2.Value t2Value

FROM dbo.Table1 t1

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

WHERE t2.ID IS NULL

GO

## RIGHT EXCLUDING JOIN

USE Fsoft\_Training

GO

SELECT t1.ID t1ID, t1.Value t1Value,t2.ID t2ID, t2.Value t2Value

FROM dbo.Table1 t1

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

WHERE t1.ID IS NULL

## OUTER JOIN EXCLUDING JOIN

USE Fsoft\_Training

GO

SELECT t1.ID t1ID, t1.Value t1Value,t2.ID t2ID, t2.Value t2Value

FROM dbo.Table1 t1

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

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

## JOINING 3 OR MORE TABLES

USE AdventureWorks

GO

SELECT p.Name, v.Name

FROM Production.Product p

JOIN Purchasing.ProductVendor pv

ON p.ProductID = pv.ProductID

JOIN Purchasing.Vendor v

ON pv.VendorID = v.VendorID

WHERE ProductSubcategoryID = 15

ORDER BY v.Name

# Subquery

## Subqueries with Aliases

This demo finds employees who have the same manager as Terri Duffy (EmployeeID = 12) by using a subquery, we should use alias for HumanResources.Employee to more clearer.

USE AdventureWorks;

GO

SELECT e1.EmployeeID,e1.LoginID, e1.ManagerID

FROM HumanResources.Employee AS e1

WHERE e1.ManagerID IN

(SELECT e2.ManagerID

FROM HumanResources.Employee AS e2

WHERE e2.EmployeeID = 12)

## Subqueries with IN

The following query finds the names of all the wheel products that Adventure Works Cycles makes.

USE AdventureWorks;

GO

SELECT Name

FROM Production.Product

WHERE ProductSubcategoryID IN

(SELECT ProductSubcategoryID

FROM Production.ProductSubcategory

WHERE Name = 'Wheels')

## Subqueries with NOT IN

The following query finds the names of the products that are not finished bicycles.

USE AdventureWorks;

GO

SELECT Name

FROM Production.Product

WHERE ProductSubcategoryID NOT IN

(SELECT ProductSubcategoryID

FROM Production.ProductSubcategory

WHERE Name = 'Mountain Bikes'

OR Name = 'Road Bikes'

OR Name = 'Touring Bikes')

## Subqueries in UPDATE

The following example doubles the value in the ListPrice column in the Production.Product table. The subquery in the WHERE clause references the Purchasing.ProductVendor table to restrict the rows updated in the Product table to just those supplied by VendorID 51.

USE AdventureWorks;

GO

UPDATE Production.Product

SET ListPrice = ListPrice \* 2

WHERE ProductID IN

(SELECT ProductID

FROM Purchasing.ProductVendor

WHERE VendorID = 51);

GO

## Subqueries with EXISTS

The following query finds the names of all products that are in the Wheels subcategory:

USE AdventureWorks;

GO

SELECT Name

FROM Production.Product

WHERE EXISTS

(SELECT \*

FROM Production.ProductSubcategory

WHERE ProductSubcategoryID =

Production.Product.ProductSubcategoryID

AND Name = 'Wheels')

## Subqueries with NOT EXISTS

For example, to find the names of products that are not in the wheels subcategory:

USE AdventureWorks;

GO

SELECT Name

FROM Production.Product

WHERE NOT EXISTS

(SELECT \*

FROM Production.ProductSubcategory

WHERE ProductSubcategoryID =

Production.Product.ProductSubcategoryID

AND Name = 'Wheels')

# Common Table Expression

## Recursive CTE

 The recursive CTE, DirectReports, defines one anchor member and one recursive member.

 The anchor member returns the base result set T0. This is the highest ranking employee in the company; that is, an employee who does not report to a manager.

 The recursive member returns the direct subordinate(s) of the employee in the anchor member result set. This is achieved by a join operation between the Employee table and the DirectReports CTE. It is this reference to the CTE itself that establishes the recursive invocation. Based on the employee in the CTE DirectReports as input (Ti), the join (Employee.ManagerID = DirectReports.EmployeeID) returns as output (Ti+1), the employees who have (Ti) as their manager

 The recursive member is activated repeatedly. The second iteration of the recursive member uses the single-row result set in step 3 (containing EmployeeID12) as the input value,

USE AdventureWorks;

GO

WITH DirectReports (ManagerID, EmployeeID, Title, DeptID, Level)

AS

(

-- Anchor member definition

SELECT e.ManagerID, e.EmployeeID, e.Title, edh.DepartmentID,

0 AS Level

FROM HumanResources.Employee AS e

INNER JOIN HumanResources.EmployeeDepartmentHistory AS edh

ON e.EmployeeID = edh.EmployeeID AND edh.EndDate IS NULL

WHERE ManagerID IS NULL

UNION ALL

-- Recursive member definition

SELECT e.ManagerID, e.EmployeeID, e.Title, edh.DepartmentID,

Level + 1

FROM HumanResources.Employee AS e

INNER JOIN HumanResources.EmployeeDepartmentHistory AS edh

ON e.EmployeeID = edh.EmployeeID AND edh.EndDate IS NULL

INNER JOIN DirectReports AS d

ON e.ManagerID = d.EmployeeID

)

-- Statement that executes the CTE

SELECT ManagerID, EmployeeID, Title, Level

FROM DirectReports

INNER JOIN HumanResources.Department AS dp

ON DirectReports.DeptID = dp.DepartmentID

WHERE dp.GroupName = N'Research and Development' OR Level = 0;

GO

# RANK function

## ROW\_NUMBER

This demo returns rows with numbers 50 to 60 in the order of the OrderDate.

USE AdventureWorks;

GO

WITH OrderedOrders AS

(

SELECT SalesOrderID, OrderDate,

ROW\_NUMBER() OVER (ORDER BY OrderDate) AS 'RowNumber'

FROM Sales.SalesOrderHeader

)

SELECT \*

FROM OrderedOrders

WHERE RowNumber BETWEEN 50 AND 60;

## RANK

The following example ranks the products in inventory according to their quantities. Each Product in an inventory will be ranked by its quantity

USE AdventureWorks;

GO

SELECT i.ProductID, p.Name, i.LocationID, i.Quantity

,RANK() OVER

(PARTITION BY i.LocationID ORDER BY i.Quantity DESC) AS 'RANK'

FROM Production.ProductInventory i

INNER JOIN Production.Product p

ON i.ProductID = p.ProductID

ORDER BY LocationID,Quantity desc;

GO

## DENSE\_RANK

The following example returns the DENSE\_RANK of the quantity of products at the various locations

USE AdventureWorks;

GO

SELECT i.ProductID, p.Name, i.LocationID, i.Quantity,

DENSE\_RANK() OVER (PARTITION BY i.LocationID ORDER BY i.Quantity) AS DENSE\_RANK

FROM Production.ProductInventory i

INNER JOIN Production.Product p

ON i.ProductID = p.ProductID

ORDER BY LocationID, Quantity DESC;

GO

## NTITLE

The rows are first partitioned by PostalCode and then divided into four groups within each PostalCode

USE AdventureWorks;

GO

SELECT c.FirstName, c.LastName

,NTILE(4) OVER(PARTITION BY PostalCode ORDER BY SalesYTD DESC) AS 'Quartile'

,s.SalesYTD, a.PostalCode

FROM Sales.SalesPerson s

INNER JOIN Person.Contact c

ON s.SalesPersonID = c.ContactID

INNER JOIN Person.Address a

ON a.AddressID = c.ContactID

WHERE TerritoryID IS NOT NULL

AND SalesYTD <> 0

ORDER BY LastName;

GO

# SQL Code Practice

Copy and **execute** below statement. Use **Include Execution Plan.** The statement use function in WHERE cost 98% comparative 2% when use don't use function

USE AdventureWorks;

GO

SELECT EmailAddress

FROM person.contact

WHERE left(EmailAddress,2) = 'As'

SELECT EmailAddress

FROM person.contact

WHERE EmailAddress like 'As%'

