(T5)比較Join和UNION。比較ISNULL、CaseWhen、COALESCE  
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc  
=======================================================================  
(T5)比較Join和UNION。比較ISNULL、CaseWhen、COALESCE  
=======================================================================  
0. In Summary

-----------

1. Query - Cross/Inner/(Left/Right/Full) Outter/Self Join Joins

1.1. (INNER)JOIN

1.2. LEFT (OUTER) JOIN

1.3. RIGHT (OUTER) JOIN

1.4. FULL (OUTER) JOIN

1.5. CROSS JOIN

1.6. SelfJoin\_LEFT/Right (Outer) Join

-----------

2. ISNULL(A,B) V.S. CaseWhen V.S. COALESCE

3. Union(All)  
=======================================================================

0. In Summary

In Summary

1. JOIN

1.1.

- (INNER) JOIN

Returns only the matching rows.

Non matching rows are eliminated.

1.2.

- LEFT (OUTER) JOIN

- LEFT (OUTER) JOIN - (INNER) JOIN

Returns all the matching rows +

non matching rows from the left table

1.3.

- RIGHT (OUTER) JOIN

- RIGHT (OUTER) JOIN - (INNER) JOIN

Returns all the matching rows +

non matching rows from the right table

1.4.

- FULL (OUTER) JOIN

- FULL (OUTER) JOIN - (INNER) JOIN

Returns all rows from both tables,

including the non-matching rows.

1.5.

- CROSS JOIN

Returns Cartesian product of the tables

involved in the join

CROSS JOIN does not need ON

1.6.

- Best SelfJoin\_LEFT/Right (Outer) Join

- 2nd-Best SelfJoin\_(INNER) JOIN

- Worst SelfJoin\_CROSS Join - No sense

---------------------------------------------

2.

2.1.

-ISNULL(A,B)

if A is NULL then B, if A is not NULL then A.

E.g.

--SELECT  ISNULL(NULL, 'No Manager') AS ManagerFullName;

--SELECT  ISNULL('Name1', 'No Manager') AS ManagerFullName;

--SELECT  COALESCE(NULL, 'No Manager') AS ManagerFullName;

--SELECT  COALESCE('Name1', 'No Manager') AS ManagerFullName;

2.2.

-CASE WHEN Expression THEN 'A' ELSE 'B' END

if expression is true, then A otherwise B

E.g.

--CASE WHEN ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ) IS NULL

--    THEN 'No Manager'

--    ELSE ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName )

--END

2.3.

-COALESCE(A, B, C ...etc)

Return the first non-NULL value.

The way to remember, COALESCE, is "Coal Ese E".

E.g.

--SELECT  COALESCE(NULL, 'A', 'B') AS FirstAvailableLeader;

Return A

--SELECT  COALESCE(NULL, NULL, 'B', 'A') AS FirstAvailableLeader;

Return B

--SELECT  COALESCE('D', NULL, 'B', 'C') AS FirstAvailableLeader;

Return D

---------------------------------------------

3.

UNION V.S. UNION ALL

3.1.

UNION removes duplicate rows,

UNION ALL does not.

3.2.

ORDER BY caluse can only be used on the last SELECT statement.

ORDER BY caluse is on any other SELECT statement will cause Syntax Error.

UNION combines rows from 2 or more tables/Search Results.

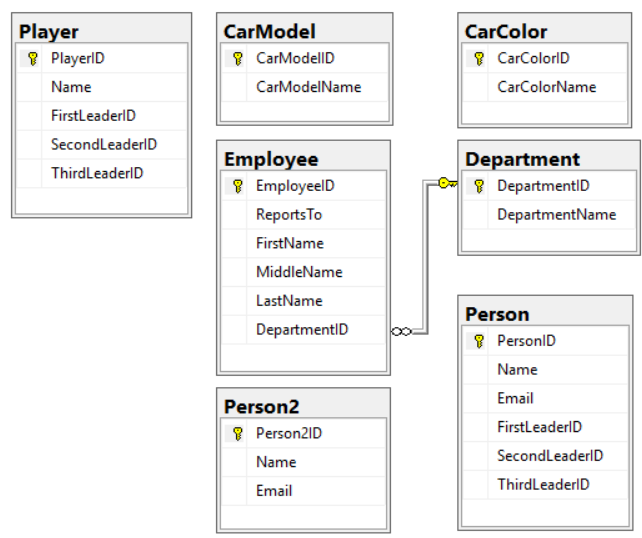
Thus, ORDER BY caluse can only be used after all the results is combined.

3.3.

UNION combines rows from 2 or more tables/Search Results.

JOINS combine columns from 2 or more tables.

---------------------------------------------



=========================================================================

1. Query - Cross/Inner/(Left/Right/Full) Outter/Self Join Joins

What to learn in this part

1. JOIN

1.1.

- (INNER) JOIN

Returns only the matching rows.

Non matching rows are eliminated.

1.2.

- LEFT (OUTER) JOIN

- LEFT (OUTER) JOIN - (INNER) JOIN

Returns all the matching rows +

non matching rows from the left table

1.3.

- RIGHT (OUTER) JOIN

- RIGHT (OUTER) JOIN - (INNER) JOIN

Returns all the matching rows +

non matching rows from the right table

1.4.

- FULL (OUTER) JOIN

- FULL (OUTER) JOIN - (INNER) JOIN

Returns all rows from both tables,

including the non-matching rows.

1.5.

- CROSS JOIN

Returns Cartesian product of the tables

involved in the join

CROSS JOIN does not need ON

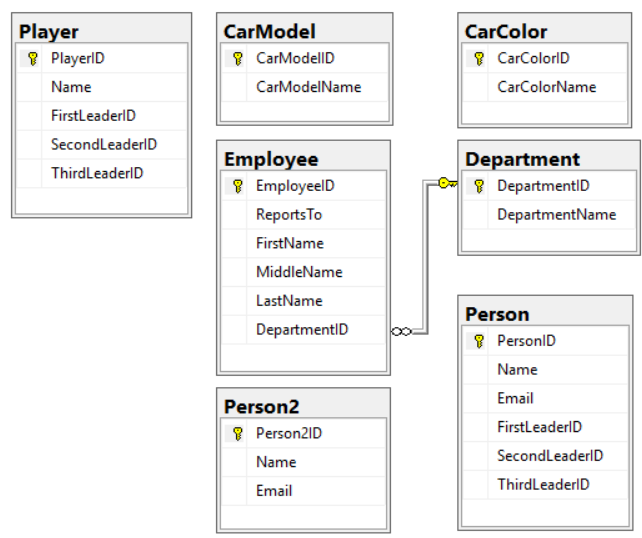
1.6.

- Best SelfJoin\_LEFT/Right (Outer) Join

- 2nd-Best SelfJoin\_(INNER) JOIN

- Worst SelfJoin\_CROSS Join - No sense

The Sample Database diagram as the following



The sample database script as the following.

--=============================================================================

--T005\_01\_(INNER)JOIN

--=============================================================================

--=============================================================================

--T005\_01\_01

--Create Sample Data

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Employee' ) )

    BEGIN

        TRUNCATE TABLE Employee;

        DROP TABLE Employee;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Department' ) )

    BEGIN

        TRUNCATE TABLE Department;

        DROP TABLE Department;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'CarModel' ) )

    BEGIN

        TRUNCATE TABLE CarModel;

        DROP TABLE CarModel;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'CarColor' ) )

    BEGIN

        TRUNCATE TABLE CarColor;

        DROP TABLE CarColor;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Player' ) )

    BEGIN

        TRUNCATE TABLE Player;

        DROP TABLE Player;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Person' ) )

    BEGIN

        TRUNCATE TABLE Person;

        DROP TABLE Person;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Person2' ) )

    BEGIN

        TRUNCATE TABLE Person2;

        DROP TABLE Person2;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE Department

(

  DepartmentID INT IDENTITY(1, 1)

                   PRIMARY KEY

                   NOT NULL ,

  DepartmentName NVARCHAR(50) NULL,

);

GO -- Run the previous command and begins new batch

INSERT  [dbo].[Department]

VALUES  ( N'Department1' );

INSERT  [dbo].[Department]

VALUES  ( N'Department2' );

INSERT  [dbo].[Department]

VALUES  ( N'Department3' );

INSERT  [dbo].[Department]

VALUES  ( N'Department4' );

INSERT  [dbo].[Department]

VALUES  ( N'Department5' );

INSERT  [dbo].[Department]

VALUES  ( N'Department6' );

GO -- Run the previous command and begins new batch

CREATE TABLE Employee

(

  EmployeeID INT IDENTITY(1, 1)

                 PRIMARY KEY

                 NOT NULL ,

  ReportsTo INT NULL ,

  FirstName NVARCHAR(100) NULL ,

  MiddleName NVARCHAR(100) NULL ,

  LastName NVARCHAR(100) NULL ,

  DepartmentID INT FOREIGN KEY REFERENCES Department ( DepartmentID )

                   NULL

);

GO -- Run the previous command and begins new batch

INSERT  Employee

VALUES  ( NULL, N'First1', N'Middle1', N'Last1', 1 );

INSERT  Employee

VALUES  ( 1, N'First2', N'Middle2', N'Last2', 2 );

INSERT  Employee

VALUES  ( 1, N'Fisrt3', N'Middle3', N'Last3', 3 );

INSERT  Employee

VALUES  ( 2, N'First4', N'Middle4', N'Last4', 1 );

INSERT  Employee

VALUES  ( 2, N'First5', N'Middle5', N'Last5', 2 );

INSERT  Employee

VALUES  ( 2, N'First6', N'Middle6', N'Last6', 3 );

INSERT  Employee

VALUES  ( 3, N'First7', N'Middle7', N'Last7', 1 );

INSERT  Employee

VALUES  ( 3, N'First8', N'Middle8', N'Last8', 2 );

INSERT  Employee

VALUES  ( 3, N'First9', N'Middle9', N'last9', NULL );

INSERT  Employee

VALUES  ( NULL, N'First10', N'Middle10', N'Last10', NULL );

GO -- Run the previous command and begins new batch

CREATE TABLE CarColor

(

  CarColorID INT IDENTITY(1, 1)

                 PRIMARY KEY

                 NOT NULL ,

  CarColorName NVARCHAR(100) NULL,

);

GO -- Run the previous command and begins new batch

INSERT  CarColor

VALUES  ( N'Green' );

INSERT  CarColor

VALUES  ( N'Blue' );

INSERT  CarColor

VALUES  ( N'Red' );

GO -- Run the previous command and begins new batch

CREATE TABLE CarModel

(

  CarModelID INT IDENTITY(1, 1)

                 PRIMARY KEY

                 NOT NULL ,

  CarModelName NVARCHAR(100) NULL,

);

GO -- Run the previous command and begins new batch

INSERT  CarModel

VALUES  ( N'Toyota Yaris' );

INSERT  CarModel

VALUES  ( N'Toyota Corolla' );

INSERT  CarModel

VALUES  ( N'Toyota Camry' );

GO -- Run the previous command and begins new batch

CREATE TABLE Player

(

  PlayerID INT IDENTITY(1, 1)

               PRIMARY KEY

               NOT NULL ,

  [Name] NVARCHAR(100) NULL ,

  FirstLeaderID INT NULL ,

  SecondLeaderID INT NULL ,

  ThirdLeaderID INT NULL,

);

GO -- Run the previous command and begins new batch

INSERT  Player

VALUES  ( N'Name1', NULL, NULL, NULL );

INSERT  Player

VALUES  ( N'Name2', NULL, 1, NULL );

INSERT  Player

VALUES  ( N'Name3', NULL, 1, 2 );

INSERT  Player

VALUES  ( N'Name4', 1, 2, 3 );

INSERT  Player

VALUES  ( N'Name5', NULL, NULL, 1 );

INSERT  Player

VALUES  ( N'Name6', NULL, 2, 3 );

INSERT  Player

VALUES  ( N'Name7', NULL, NULL, 3 );

INSERT  Player

VALUES  ( N'Name8', NULL, 1, 2 );

INSERT  Player

VALUES  ( N'Name9', 1, 2, 3 );

INSERT  Player

VALUES  ( N'Name10', NULL, 1, 2 );

GO -- Run the previous command and begins new batch

CREATE TABLE Person

(

  PersonID INT IDENTITY(1, 1)

               PRIMARY KEY

               NOT NULL ,

  [Name] NVARCHAR(100) NULL ,

  Email NVARCHAR(500) NULL ,

  FirstLeaderID INT NULL ,

  SecondLeaderID INT NULL ,

  ThirdLeaderID INT NULL,

);

GO -- Run the previous command and begins new batch

INSERT  Person

VALUES  ( N'Name1', N'[1@1.com](mailto:1@1.com)', NULL, NULL, NULL );

INSERT  Person

VALUES  ( N'Name2', N'[2@2.com](mailto:2@2.com)', NULL, 1, NULL );

INSERT  Person

VALUES  ( N'Name3', N'[3@3.com](mailto:3@3.com)', NULL, 1, 2 );

INSERT  Person

VALUES  ( N'Name4', N'[4@4.com](mailto:4@4.com)', 1, 2, 3 );

INSERT  Person

VALUES  ( N'Name5', N'[5@5.com](mailto:5@5.com)', NULL, NULL, 1 );

INSERT  Person

VALUES  ( N'Name6', N'[6@6.com](mailto:6@6.com)', NULL, 2, 3 );

INSERT  Person

VALUES  ( N'Name7', N'[7@7.com](mailto:7@7.com)', NULL, NULL, 3 );

INSERT  Person

VALUES  ( N'Name8', N'[8@8.com](mailto:8@8.com)', NULL, 1, 2 );

INSERT  Person

VALUES  ( N'Name9', N'[9@9.com](mailto:9@9.com)', 1, 2, 3 );

INSERT  Person

VALUES  ( N'Name10', N'[10@10.com](mailto:10@10.com)', NULL, 1, 2 );

GO -- Run the previous command and begins new batch

CREATE TABLE Person2

(

  Person2ID INT IDENTITY(1, 1)

                PRIMARY KEY

                NOT NULL ,

  [Name] NVARCHAR(100) NULL ,

  Email NVARCHAR(500) NULL,

);

GO -- Run the previous command and begins new batch

INSERT  Person2

VALUES  ( N'Name6', N'[6@6.com](mailto:6@6.com)' );

INSERT  Person2

VALUES  ( N'Name7', N'[7@7.com](mailto:7@7.com)' );

INSERT  Person2

VALUES  ( N'Name8', N'[8@8.com](mailto:8@8.com)' );

INSERT  Person2

VALUES  ( N'Name9', N'[9@9.com](mailto:9@9.com)' );

INSERT  Person2

VALUES  ( N'Name10', N'[10@10.com](mailto:10@10.com)' );

INSERT  Person2

VALUES  ( N'Name11', N'[11@11.com](mailto:11@11.com)' );

INSERT  Person2

VALUES  ( N'Name12', N'[12@12.com](mailto:12@12.com)' );

INSERT  Person2

VALUES  ( N'Name13', N'[13@13.com](mailto:13@13.com)' );

INSERT  Person2

VALUES  ( N'Name14', N'[14@14.com](mailto:14@14.com)' );

INSERT  Person2

VALUES  ( N'Name15', N'[15@15.com](mailto:15@15.com)' );

GO -- Run the previous command and begins new batch

SELECT  \*

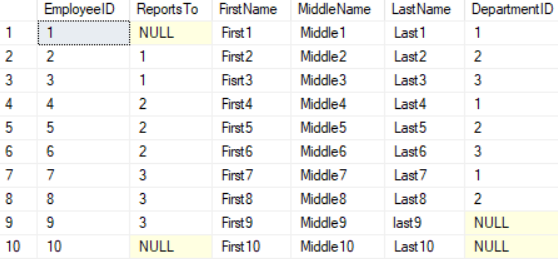
FROM    Department;

Table

Description automatically generated

SELECT  \*

FROM    Employee;



SELECT  \*

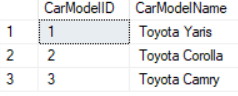
FROM    CarColor;

Table

Description automatically generated

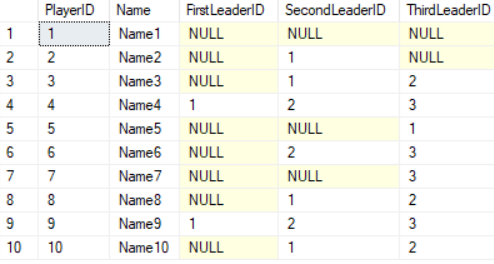
SELECT  \*

FROM    CarModel;



SELECT  \*

FROM    Player;



SELECT  \*

FROM    Person;

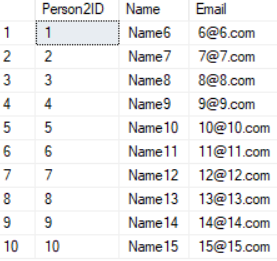
Table

Description automatically generated

SELECT  \*

FROM    Person2;

GO -- Run the previous command and begins new batch



=========================================================================

1.1. (INNER)JOIN

--=============================================================================

--T005\_01\_02

--(INNER) JOIN

SELECT  \*

FROM    Employee;

SELECT  \*

FROM    Department;

SELECT  ( e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName ) AS FullName ,

        d.DepartmentName

FROM    dbo.Employee e

        INNER JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID;

        -- JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID;

GO -- Run the prvious command and begins new batch

/\*

1.

(INNER) JOIN

Returns only the matching rows.

Non matching rows are eliminated.

2.

Employee has 10 rows

Department has 6 rows

There are 8 matching rows.

There are 2 non-matching rows from Employee.

There are 3 non-matching rows from Department.

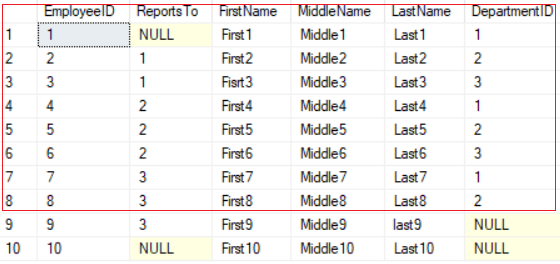
-->

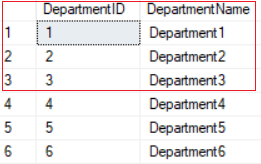
(INNER) JOIN will only show 8 matching rows.

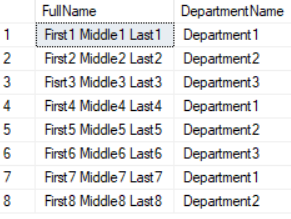
\*/

Text

Description automatically generated with low confidence







=========================================================================

1.2. LEFT (OUTER) JOIN

--=============================================================================

--T005\_02\_LEFT (OUTER) JOIN

--=============================================================================

--=============================================================================

--T005\_02\_01

--LEFT (OUTER) JOIN

SELECT  \*

FROM    Employee;

SELECT  \*

FROM    Department;

SELECT  ( e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName ) AS FullName ,

        d.DepartmentName

FROM    dbo.Employee e

        LEFT OUTER JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID;

        --LEFT JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID;

GO -- Run the prvious command and begins new batch

/\*

1.

LEFT (OUTER) JOIN

Returns all the matching rows +

non matching rows from the left table

2.

Employee has 10 rows

Department has 6 rows

There are 8 matching rows.

There are 2 non-matching rows from Employee.

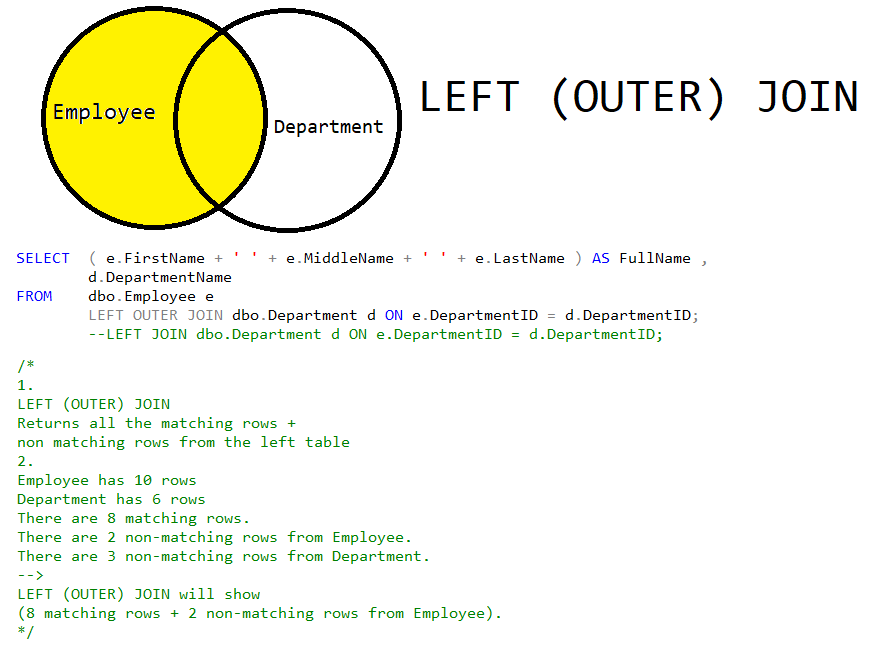
There are 3 non-matching rows from Department.

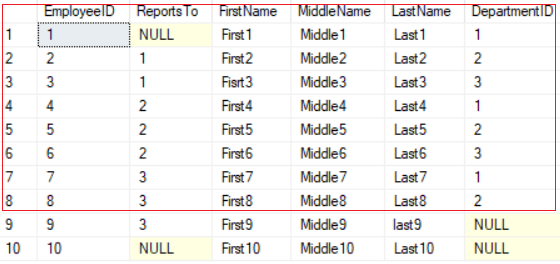
-->

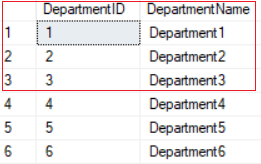
LEFT (OUTER) JOIN will show

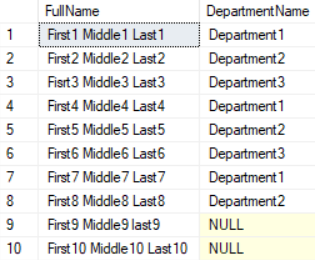
(8 matching rows + 2 non-matching rows from Employee).

\*/









--=============================================================================

--T005\_02\_02

--LEFT (OUTER) JOIN - (INNER) JOIN

USE Sample;

GO -- Run the prvious command and begins new batch

SELECT  \*

FROM    Employee;

SELECT  \*

FROM    Department;

SELECT  ( e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName ) AS FullName ,

        d.DepartmentName

FROM    dbo.Employee e

        LEFT OUTER JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID

             --LEFT JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID;

WHERE   d.DepartmentID IS NULL;

GO -- Run the prvious command and begins new batch

/\*

1.

LEFT (OUTER) JOIN

Returns all the matching rows +

non matching rows from the left table

2.

Employee has 10 rows

Department has 6 rows

There are 8 matching rows.

There are 2 non-matching rows from Employee.

There are 3 non-matching rows from Department.

-->

LEFT (OUTER) JOIN will show

(8 matching rows + 2 non-matching rows from Employee).

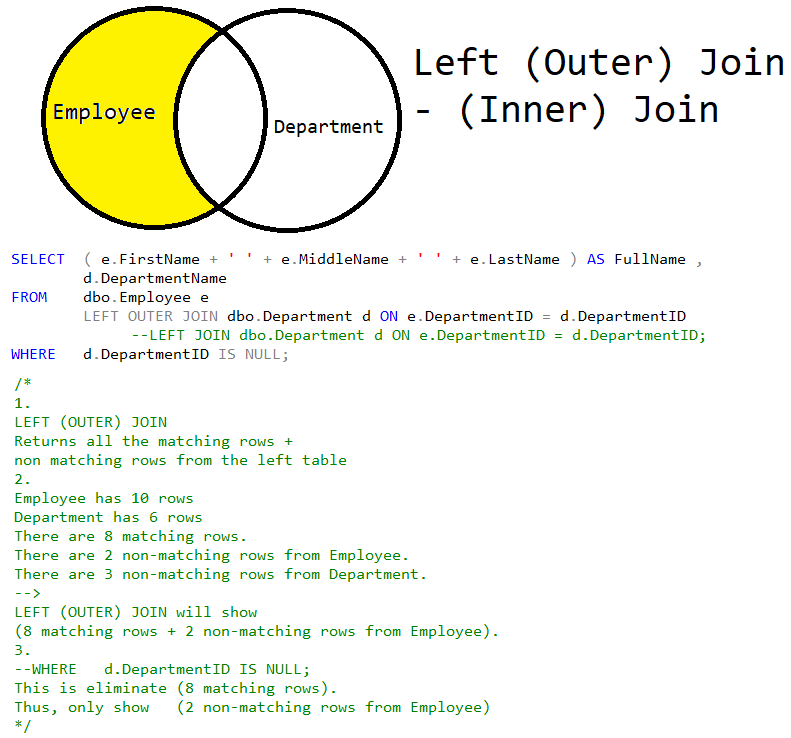
3.

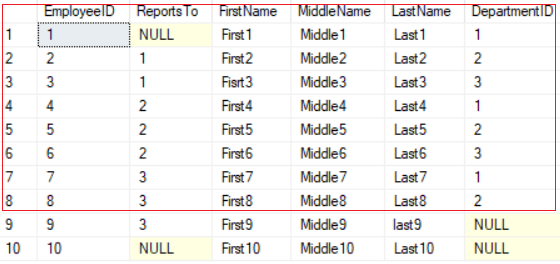
--WHERE   d.DepartmentID IS NULL;

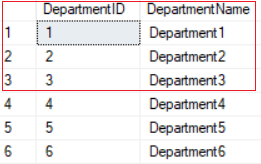
This is eliminate (8 matching rows).

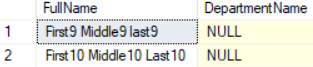
Thus, only show   (2 non-matching rows from Employee)

\*/









=========================================================================

1.3. RIGHT (OUTER) JOIN

--=============================================================================

--T005\_03\_RIGHT (OUTER) JOIN

--=============================================================================

--=============================================================================

--T005\_03\_01

--RIGHT (OUTER) JOIN

SELECT  \*

FROM    Employee;

SELECT  \*

FROM    Department;

SELECT  ( e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName ) AS FullName ,

        d.DepartmentName

FROM    dbo.Employee e

        RIGHT OUTER JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID;

        --RIGHT JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID;

GO -- Run the prvious command and begins new batch

/\*

1.

--RIGHT (OUTER) JOIN

Returns all the matching rows +

non matching rows from the right table

2.

Employee has 10 rows

Department has 6 rows

There are 8 matching rows.

There are 2 non-matching rows from Employee.

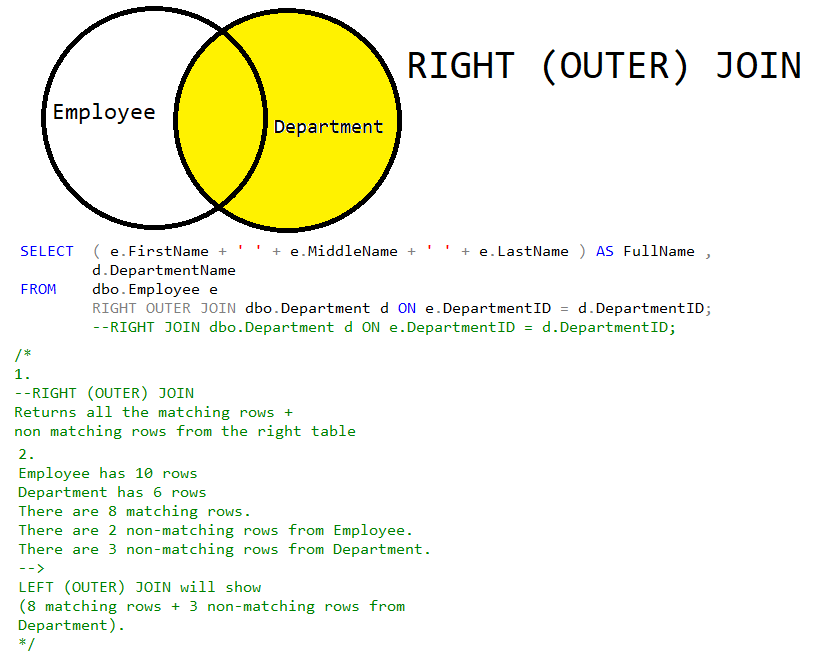
There are 3 non-matching rows from Department.

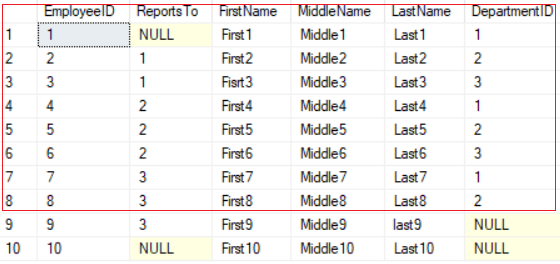
-->

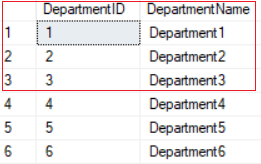
LEFT (OUTER) JOIN will show

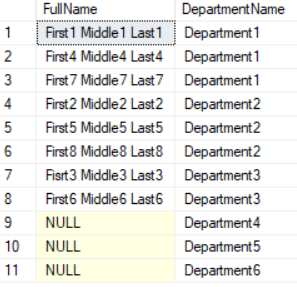
(8 matching rows + 3 non-matching rows from Department).

\*/









--=============================================================================

--T005\_03\_02

--RIGHT (OUTER) JOIN - (INNER) JOIN

SELECT  \*

FROM    Employee;

SELECT  \*

FROM    Department;

SELECT  ( e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName ) AS FullName ,

        d.DepartmentName

FROM    dbo.Employee e

        RIGHT OUTER JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID

             --RIGHT JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID;

WHERE   e.DepartmentID IS NULL;

GO -- Run the prvious command and begins new batch

--LEFT (OUTER) JOIN - INNER JOIN

/\*

1.

--RIGHT (OUTER) JOIN

Returns all the matching rows +

non matching rows from the right table

2.

Employee has 10 rows

Department has 6 rows

There are 8 matching rows.

There are 2 non-matching rows from Employee.

There are 3 non-matching rows from Department.

-->

LEFT (OUTER) JOIN will show

(8 matching rows + 3 non-matching rows from Department).

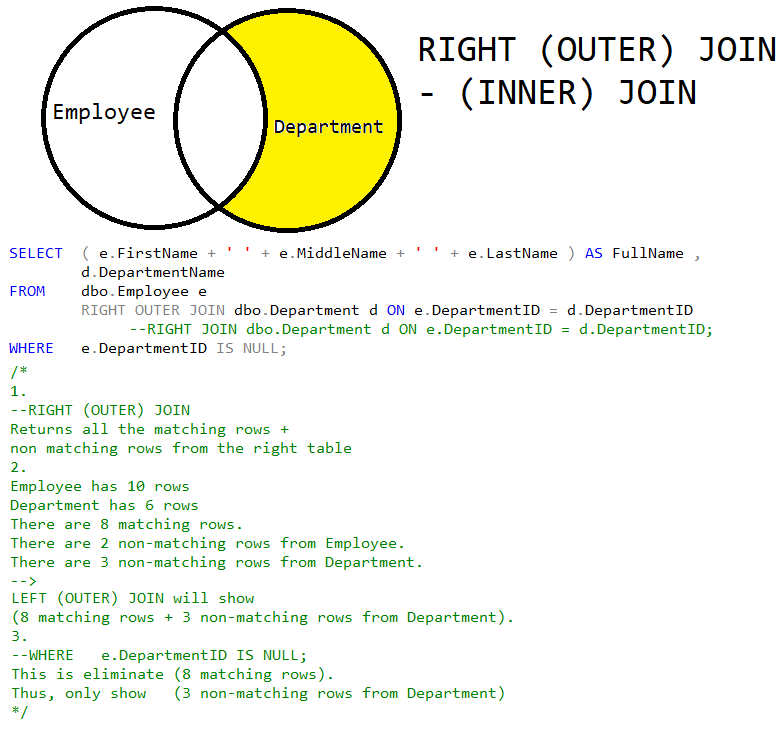
3.

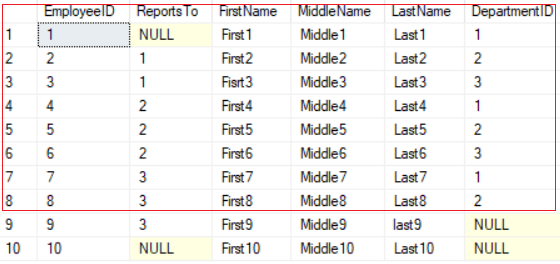
--WHERE   e.DepartmentID IS NULL;

This is eliminate (8 matching rows).

Thus, only show   (3 non-matching rows from Department)

\*/





Table

Description automatically generated

Table

Description automatically generated

=========================================================================

1.4. FULL (OUTER) JOIN

--=============================================================================

--T005\_04\_FULL (OUTER) JOIN

--=============================================================================

--=============================================================================

--T005\_04\_01

--FULL (OUTER) JOIN

SELECT  \*

FROM    Employee;

SELECT  \*

FROM    Department;

SELECT  ( e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName ) AS FullName ,

        d.DepartmentName

FROM    dbo.Employee e

        FULL OUTER JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID;

        --FULL JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID;

GO -- Run the prvious command and begins new batch

/\*

1.

--FULL (OUTER) JOIN

Returns all rows from both tables,

including the non-matching rows.

2.

Employee has 10 rows

Department has 6 rows

There are 8 matching rows.

There are 2 non-matching rows from Employee.

There are 3 non-matching rows from Department.

-->

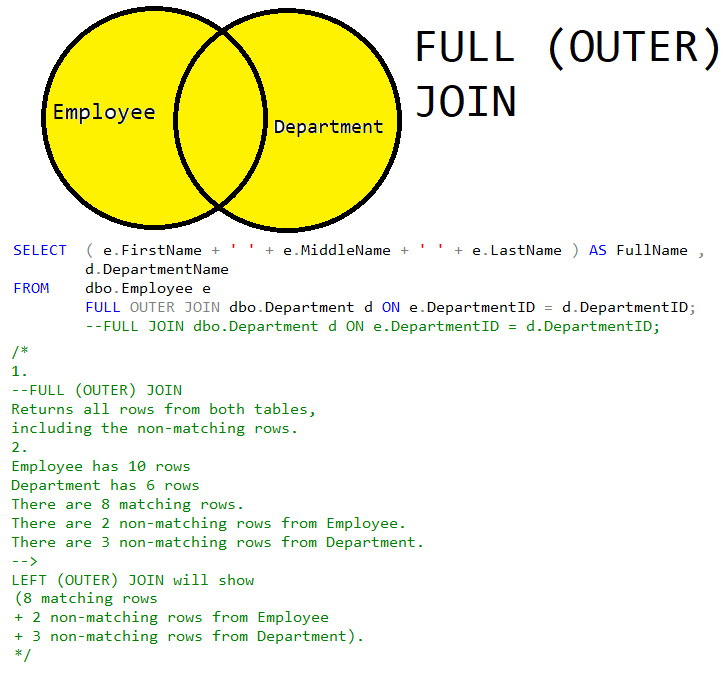
LEFT (OUTER) JOIN will show

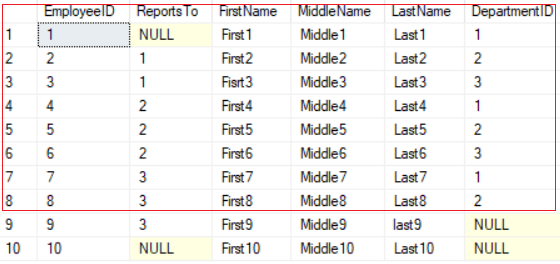
(8 matching rows

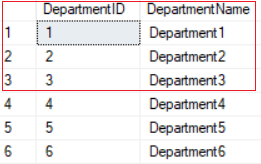
+ 2 non-matching rows from Employee

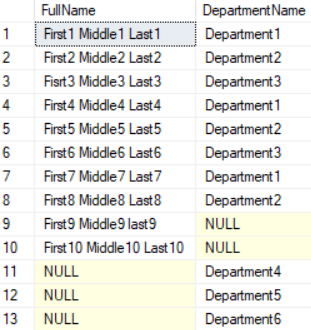
+ 3 non-matching rows from Department).

\*/









--=============================================================================

--T005\_04\_02

--FULL (OUTER) JOIN - (INNER) JOIN

USE Sample;

GO -- Run the prvious command and begins new batch

SELECT  \*

FROM    Employee;

SELECT  \*

FROM    Department;

SELECT  ( e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName ) AS FullName ,

        d.DepartmentName

FROM    dbo.Employee e

        FULL OUTER JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID

              --FULL JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID;

WHERE   e.DepartmentID IS NULL

        OR d.DepartmentID IS NULL;

GO -- Run the prvious command and begins new batch

/\*

1.

--FULL (OUTER) JOIN

Returns all rows from both tables,

including the non-matching rows.

2.

Employee has 10 rows

Department has 6 rows

There are 8 matching rows.

There are 2 non-matching rows from Employee.

There are 3 non-matching rows from Department.

-->

LEFT (OUTER) JOIN will show

(8 matching rows

+ 2 non-matching rows from Employee

+ 3 non-matching rows from Department).

3.

--WHERE   e.DepartmentID IS NULL;

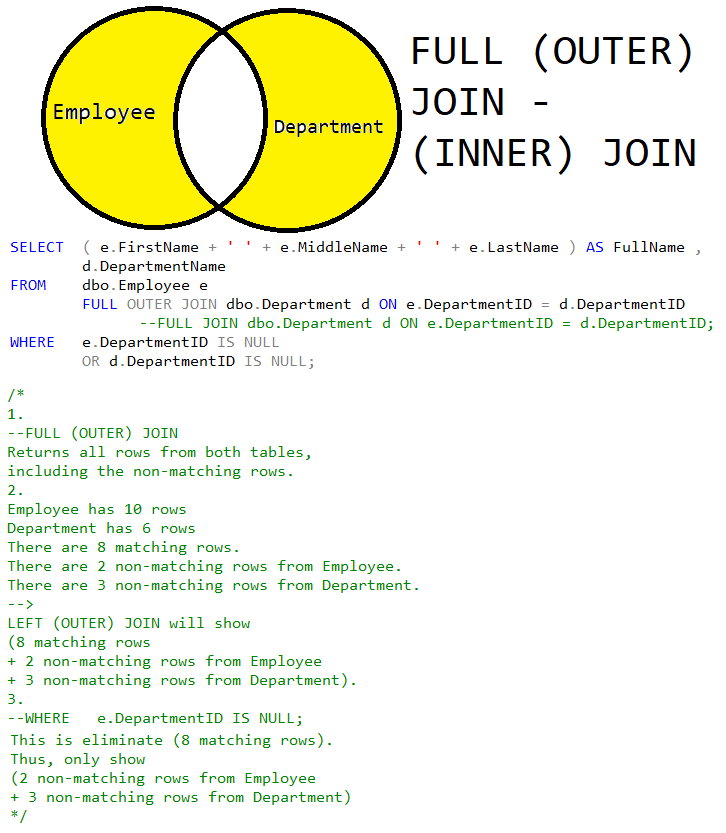
This is eliminate (8 matching rows).

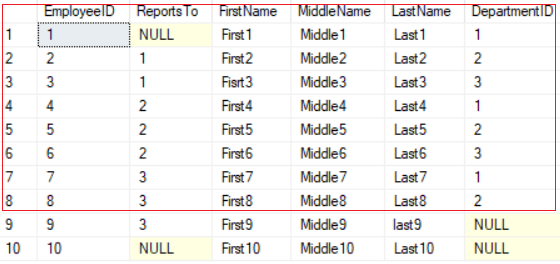
Thus, only show

(2 non-matching rows from Employee

+ 3 non-matching rows from Department)

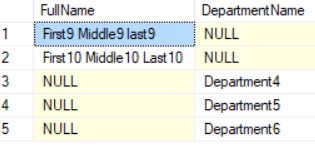
\*/





Table

Description automatically generated



=========================================================================

1.5. CROSS JOIN

--=============================================================================

--T005\_05\_CROSS JOIN

--=============================================================================

--=============================================================================

--T005\_05\_01

--CROSS JOIN

SELECT  \*

FROM    Employee;

SELECT  \*

FROM    Department;

SELECT  ( e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName ) AS FullName ,

        d.DepartmentName

FROM    dbo.Employee e

        CROSS JOIN dbo.Department d;

GO -- Run the prvious command and begins new batch

/\*

1.

-- CROSS JOIN

Returns Cartesian product of the tables

involved in the join.

CROSS JOIN does not need ON

2.

Employee has 10 rows

Department has 6 rows

There are 8 matching rows.

There are 2 non-matching rows from Employee.

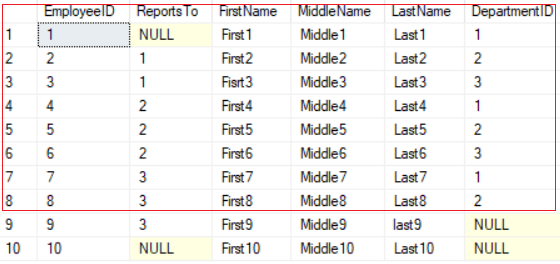
There are 3 non-matching rows from Department.

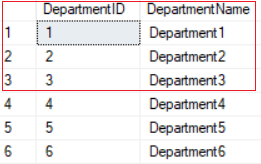
-->

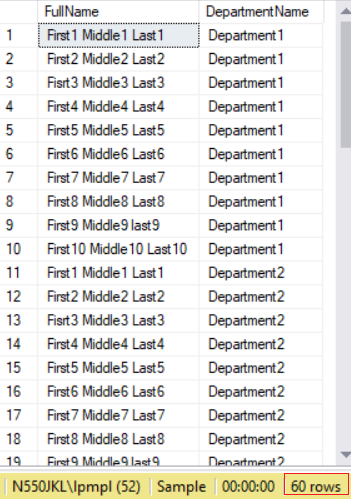
CROSS JOIN will show

(10 rows from Employee) \* (6 rows from Department) = 60 rows

\*/







--=============================================================================

--T005\_05\_02

--CROSS JOIN

SELECT  \*

FROM    CarColor;

SELECT  \*

FROM    CarModel;

SELECT  ( cm.CarModelName + ' ' + cc.CarColorName ) AS CarList

FROM    dbo.CarColor cc

        CROSS JOIN dbo.CarModel cm;

GO -- Run the prvious command and begins new batch

/\*

1.

-- CROSS JOIN

Returns Cartesian product of the tables

involved in the join.

CROSS JOIN does not need ON

2.

CarColor has 3 rows

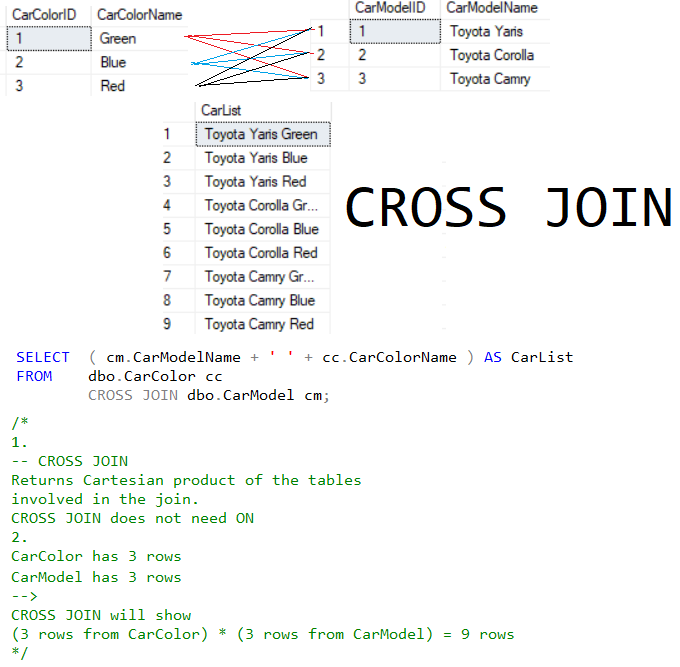
CarModel has 3 rows

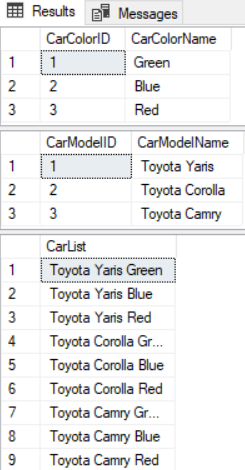
-->

CROSS JOIN will show

(3 rows from CarColor) \* (3 rows from CarModel) = 9 rows

\*/





1.6. SelfJoin\_LEFT/Right (Outer) Join

--=============================================================================

--T005\_06\_SelfJoin\_LEFT/Right (Outer) Join

--=============================================================================

--=============================================================================

--T005\_06\_01

--Best SelfJoin\_LEFT/Right (Outer) Join

SELECT  \*

FROM    Employee;

SELECT  ( e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName ) AS FullName ,

        ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ) AS ManagerFullName

FROM    dbo.Employee e

        LEFT OUTER JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

             --LEFT JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

GO -- Run the prvious command and begins new batch

/\*

1.

Best SelfJoin\_LEFT/Right (Outer) Join

2.

- LEFT (OUTER) JOIN

Returns all the matching rows +

non matching rows from the left table

3.

Left Employee e has 10 rows

Right Employee m has 10 rows

-- LEFT (OUTER) JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

There are 8 matching rows.

There are 2 non-matching rows from Left Employee e

There are 2 non-matching rows from Right Employee m

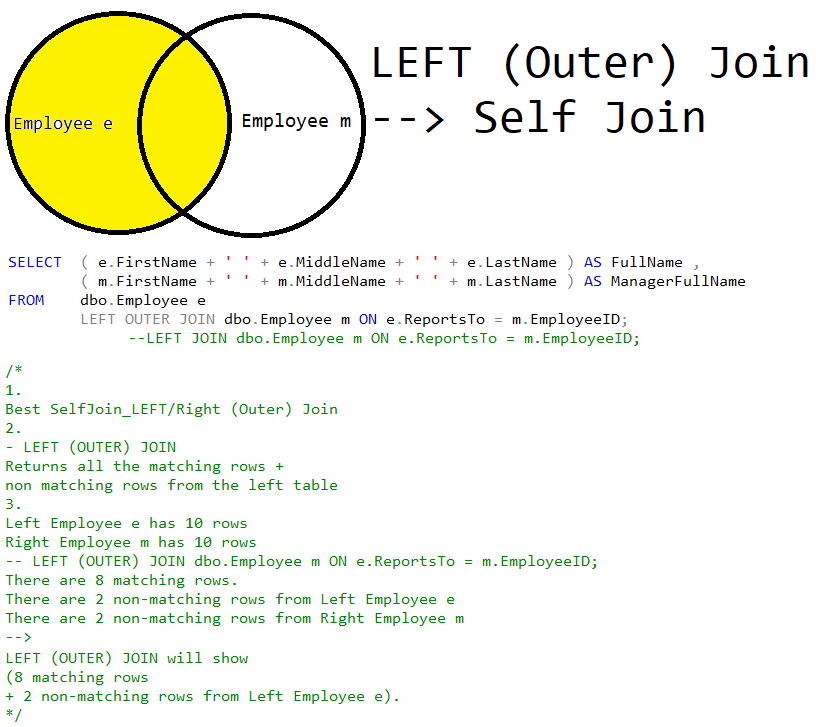
-->

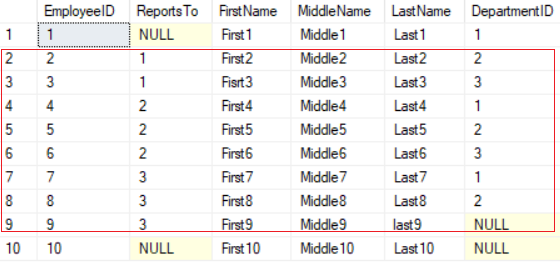
LEFT (OUTER) JOIN will show

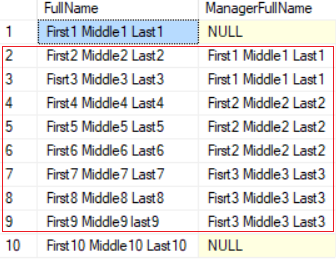
(8 matching rows

+ 2 non-matching rows from Left Employee e).

\*/







--=============================================================================

--T005\_06\_02

--2nd-Best SelfJoin\_(INNER) JOIN

SELECT  \*

FROM    Employee;

SELECT  ( e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName ) AS FullName ,

        ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ) AS ManagerFullName

FROM    dbo.Employee e

        INNER JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

             --JOIN JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

GO -- Run the prvious command and begins new batch

/\*

1.

2nd-Best SelfJoin\_(INNER) JOIN

2.

- (INNER) JOIN

Returns only the matching rows.

Non matching rows are eliminated.

3.

Left Employee e has 10 rows

Right Employee m has 10 rows

-- LEFT (OUTER) JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

There are 8 matching rows.

There are 2 non-matching rows from Left Employee e

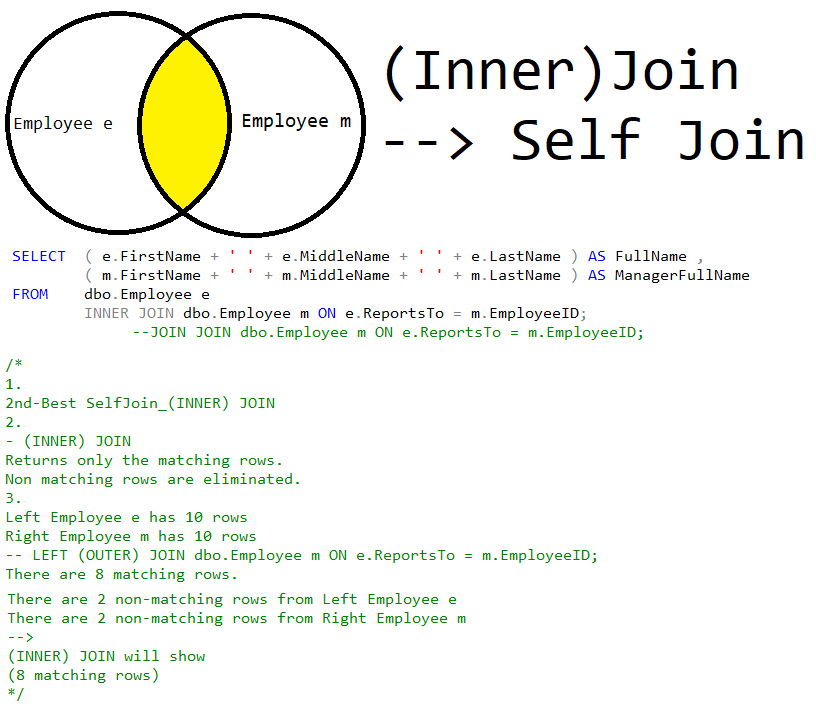
There are 2 non-matching rows from Right Employee m

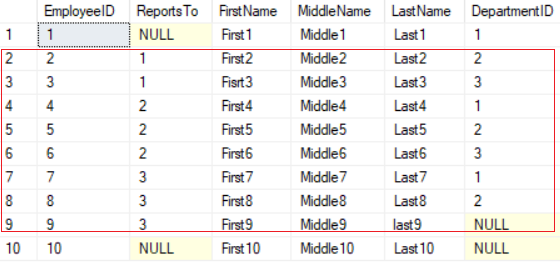
-->

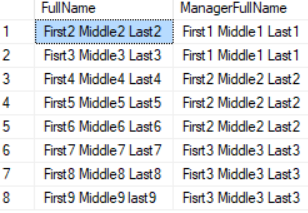
(INNER) JOIN will show

(8 matching rows)

\*/







--=============================================================================

--T005\_06\_03

--Worst SelfJoin\_CROSS Join - No sense

SELECT  \*

FROM    Employee;

SELECT  ( e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName ) AS FullName ,

        ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ) AS ManagerFullName

FROM    dbo.Employee e

        CROSS JOIN dbo.Employee m;

GO -- Run the prvious command and begins new batch

/\*

1.

Worst SelfJoin\_CROSS Join - No sense

2.

-- CROSS JOIN

Returns Cartesian product of the tables

involved in the join.

CROSS JOIN does not need ON

3.

Left Employee e has 10 rows

Right Employee m has 10 rows

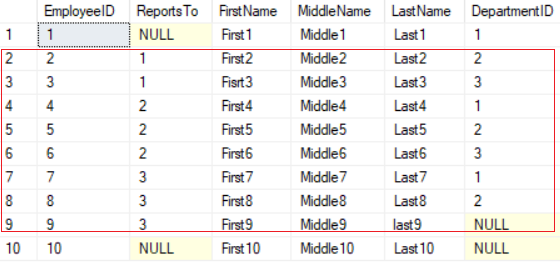
-->

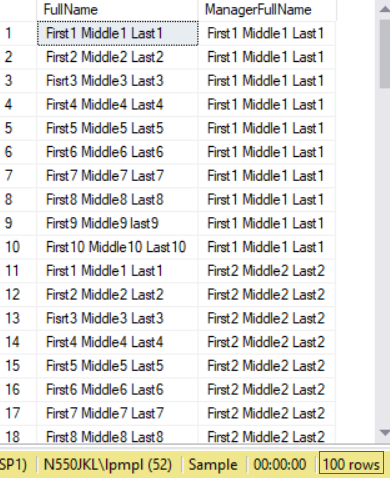
CROSS JOIN JOIN will show

(10 rows from Left Employee e) \*

(10 rows from Right Employee m) = 100 rows

\*/





2. ISNULL(A,B) V.S. CaseWhen V.S. COALESCE

--=============================================================================

--T005\_07\_ISNULL(A,B) V.S. CaseWhen V.S. COALESCE

--=============================================================================

--=============================================================================

--T005\_07\_01

--Best SelfJoin\_LEFT/Right (Outer) Join

SELECT  \*

FROM    Employee;

SELECT  ( e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName ) AS FullName ,

        ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ) AS ManagerFullName

FROM    dbo.Employee e

        LEFT OUTER JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

             --LEFT JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

GO -- Run the prvious command and begins new batch

/\*

1.

Best SelfJoin\_LEFT/Right (Outer) Join

2.

- LEFT (OUTER) JOIN

Returns all the matching rows +

non matching rows from the left table

3.

Left Employee e has 10 rows

Right Employee m has 10 rows

-- LEFT (OUTER) JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

There are 8 matching rows.

There are 2 non-matching rows from Left Employee e

There are 2 non-matching rows from Right Employee m

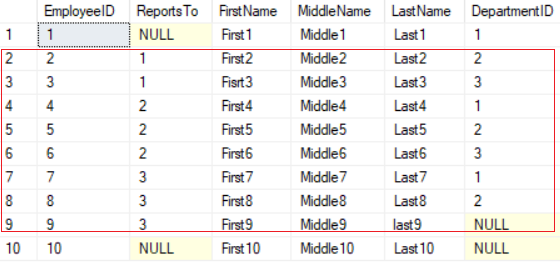
-->

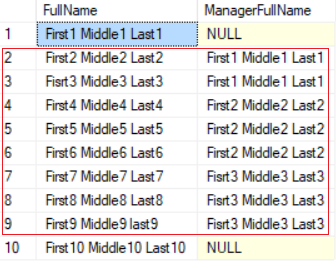
LEFT (OUTER) JOIN will show

(8 matching rows

+ 2 non-matching rows from Left Employee e).

\*/





--=============================================================================

--T005\_07\_02

--Best SelfJoin\_LEFT/Right (Outer) Join

--ISNULL(A,B)

-- if A is NULL then B, if A is not NULL then A.

SELECT  \*

FROM    Employee;

SELECT  ISNULL(NULL, 'No Manager') AS ManagerFullName;

SELECT  ISNULL('Name1', 'No Manager') AS ManagerFullName;

SELECT  ( e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName ) AS FullName ,

        ISNULL(( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ),

               'No Manager') AS ManagerFullName

        --( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ) AS ManagerFullName

FROM    dbo.Employee e

        LEFT OUTER JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

             --LEFT JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

GO -- Run the prvious command and begins new batch

/\*

1.

Best SelfJoin\_LEFT/Right (Outer) Join

2.

- LEFT (OUTER) JOIN

Returns all the matching rows +

non matching rows from the left table

3.

Left Employee e has 10 rows

Right Employee m has 10 rows

-- LEFT (OUTER) JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

There are 8 matching rows.

There are 2 non-matching rows from Left Employee e

There are 2 non-matching rows from Right Employee m

-->

LEFT (OUTER) JOIN will show

(8 matching rows

+ 2 non-matching rows from Left Employee e).

4.

--ISNULL(A,B)

if A is NULL then B, if A is not NULL then A.

--ISNULL(( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ),

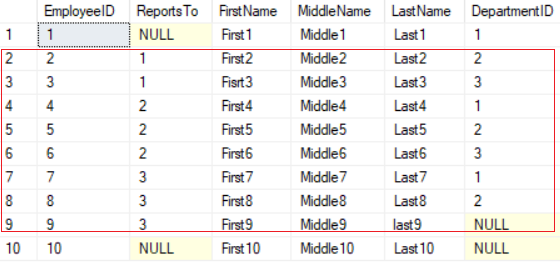
--    'No Manager') AS ManagerFullName

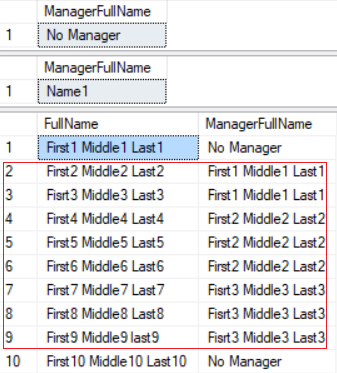
if ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ) is NULL, then 'No Manager'

if ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ) is NOT NULL, then

( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName )

\*/





--=============================================================================

--T005\_07\_03

--Best SelfJoin\_LEFT/Right (Outer) Join

--CASE WHEN Expression THEN 'A' ELSE 'B' END

--if expression is true, then A otherwise B

SELECT  \*

FROM    Employee;

SELECT  ( e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName ) AS FullName ,

        CASE WHEN ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ) IS NULL

             THEN 'No Manager'

             ELSE ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName )

        END

             --COALESCE(( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ), 'No Manager') AS ManagerFullName

             --ISNULL(( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ), 'No Manager') AS ManagerFullName

        --( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ) AS ManagerFullName

FROM    dbo.Employee e

        LEFT OUTER JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

             --LEFT JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

GO -- Run the prvious command and begins new batch

/\*

1.

Best SelfJoin\_LEFT/Right (Outer) Join

2.

- LEFT (OUTER) JOIN

Returns all the matching rows +

non matching rows from the left table

3.

Left Employee e has 10 rows

Right Employee m has 10 rows

-- LEFT (OUTER) JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

There are 8 matching rows.

There are 2 non-matching rows from Left Employee e

There are 2 non-matching rows from Right Employee m

-->

LEFT (OUTER) JOIN will show

(8 matching rows

+ 2 non-matching rows from Left Employee e).

4.

--ISNULL(A,B)

if A is NULL then B, if A is not NULL then A.

--ISNULL(( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ),

--    'No Manager') AS ManagerFullName

if ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ) is NULL, then 'No Manager'

if ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ) is NOT NULL, then

( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName )

5.

--CASE

--     WHEN Expression

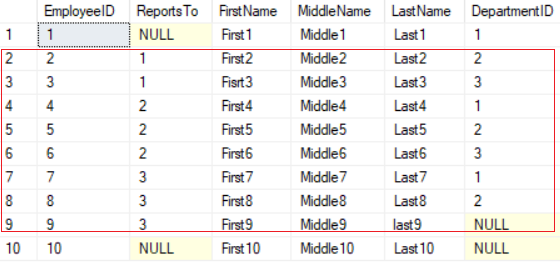
--     THEN 'A'

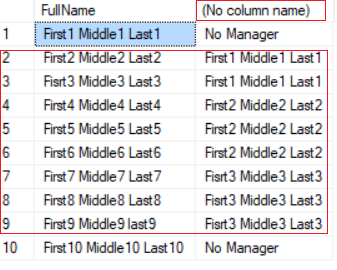
--     ELSE 'B'

--END

if expression is true, then A otherwise B

\*/





--=============================================================================

--T005\_07\_04

--Best SelfJoin\_LEFT/Right (Outer) Join

--COALESCE(A, B, C ...etc)

--Return the first non-NULL value.

SELECT  \*

FROM    Employee;

SELECT  ISNULL(NULL, 'No Manager') AS ManagerFullName;

SELECT  ISNULL('Name1', 'No Manager') AS ManagerFullName;

SELECT  COALESCE(NULL, 'No Manager') AS ManagerFullName;

SELECT  COALESCE('Name1', 'No Manager') AS ManagerFullName;

SELECT  ( e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName ) AS FullName ,

        COALESCE(( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ),

                 'No Manager') AS ManagerFullName

        --CASE WHEN ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ) IS NULL

        --     THEN 'No Manager'

        --     ELSE ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName )

        --END

             --ISNULL(( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ), 'No Manager') AS ManagerFullName

        --( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ) AS ManagerFullName

FROM    dbo.Employee e

        LEFT OUTER JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

             --LEFT JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

GO -- Run the prvious command and begins new batch

/\*

1.

Best SelfJoin\_LEFT/Right (Outer) Join

2.

- LEFT (OUTER) JOIN

Returns all the matching rows +

non matching rows from the left table

3.

Left Employee e has 10 rows

Right Employee m has 10 rows

-- LEFT (OUTER) JOIN dbo.Employee m ON e.ReportsTo = m.EmployeeID;

There are 8 matching rows.

There are 2 non-matching rows from Left Employee e

There are 2 non-matching rows from Right Employee m

-->

LEFT (OUTER) JOIN will show

(8 matching rows

+ 2 non-matching rows from Left Employee e).

4.

--ISNULL(A,B)

if A is NULL then B, if A is not NULL then A.

--ISNULL(( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ),

--    'No Manager') AS ManagerFullName

if ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ) is NULL, then 'No Manager'

if ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ) is NOT NULL, then

( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName )

5.

--CASE

--     WHEN Expression

--     THEN 'A'

--     ELSE 'B'

--END

if expression is true, then A otherwise B

6.

--COALESCE(A, B, C ...etc)

Return the first non-NULL value.

--COALESCE(( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName ),

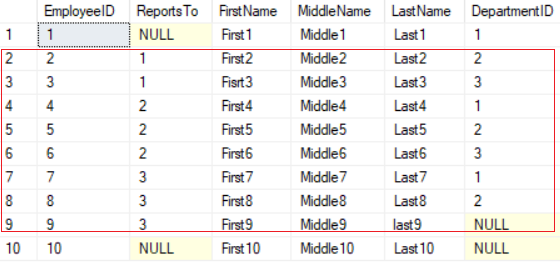
--    'No Manager') AS ManagerFullName

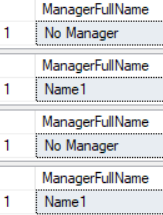
first value is ( m.FirstName + ' ' + m.MiddleName + ' ' + m.LastName )

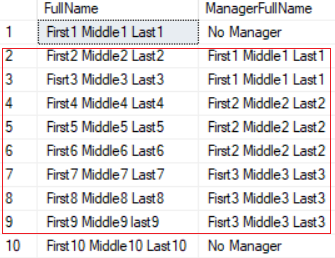
second value is  'No Manager'

Return the first non-NULL value.

\*/







--=============================================================================

--T005\_07\_05

--COALESCE(A, B, C ...etc)

--Return the first non-NULL value.

SELECT  \*

FROM    Player;

SELECT  COALESCE(NULL, 'A', 'B') AS FirstAvailableLeader;

SELECT  COALESCE(NULL, NULL, 'B', 'A') AS FirstAvailableLeader;

SELECT  COALESCE('D', NULL, 'B', 'C') AS FirstAvailableLeader;

SELECT  p.PlayerID ,

        p.Name ,

        COALESCE(p.FirstLeaderID, p.SecondLeaderID, p.ThirdLeaderID, 0) AS FirstAvailableLeaderID ,

        (

                    --SELECT ISNULL(p2.[Name], 'No Leader')

          SELECT    p2.[Name]

          FROM      dbo.Player p2

          WHERE     p2.PlayerID = COALESCE(p.FirstLeaderID, p.SecondLeaderID,

                                           p.ThirdLeaderID)

        ) AS FirstAvailableLeader

FROM    dbo.Player p;

GO -- Run the prvious command and begins new batch

/\*

1.

--COALESCE(A, B, C ...etc)

Return the first non-NULL value.

2.

--SELECT  COALESCE(NULL, 'A', 'B') AS FirstAvailableLeader;

Return A

--SELECT  COALESCE(NULL, NULL, 'B', 'A') AS FirstAvailableLeader;

Return B

--SELECT  COALESCE('D', NULL, 'B', 'C') AS FirstAvailableLeader;

Return D

3.

-- COALESCE(p.FirstLeaderID, p.SecondLeaderID, p.ThirdLeaderID, 0) AS FirstAvailableLeaderID ,

first valus is FirstLeaderID

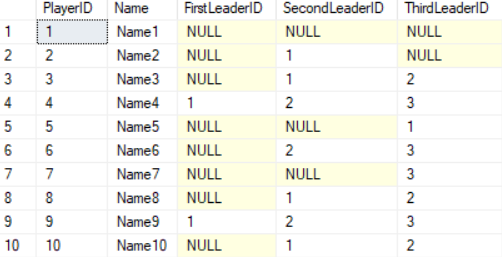
second value is SecondLeaderID

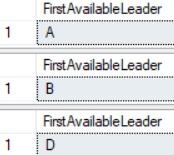
third value is ThirdLeaderID

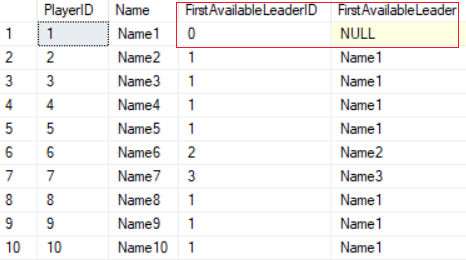
fourth value is 0

Return the first non-NULL value

\*/







==================================================================================

3. Union(All)

--=============================================================================

--T005\_08\_Union(All)

--=============================================================================

--=============================================================================

--T005\_08\_01

--UNION ALL

SELECT  \*

FROM    dbo.Person;

SELECT  \*

FROM    dbo.Person2;

SELECT  p1.[Name] ,

        p1.[Email]

FROM    dbo.Person p1

UNION ALL

SELECT  p2.[Name] ,

        p2.[Email]

FROM    dbo.Person2 p2;

GO -- Run the prvious command and begins new batch

/\*

Person has 10 rows

Person2 has 10 rows

Compare Person and Person2, there are 5 rows are the same.

-->

UNION removes duplicate rows,

UNION ALL does not.

-->

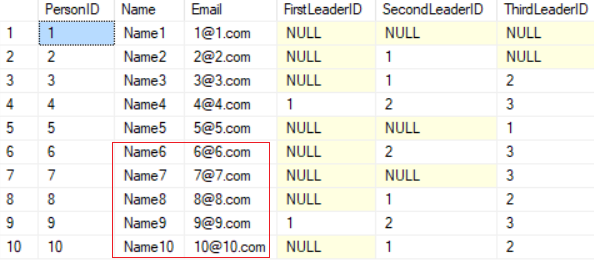
UNION ALL

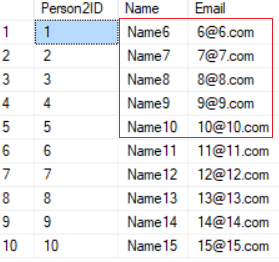
-->

(10 rows from Person

+ 10 rows from Person2) = 20 rows

\*/







--=============================================================================

--T005\_08\_02

--UNION

SELECT  \*

FROM    dbo.Person;

SELECT  \*

FROM    dbo.Person2;

SELECT  p1.[Name] ,

        p1.[Email]

FROM    dbo.Person p1

UNION

SELECT  p2.[Name] ,

        p2.[Email]

FROM    dbo.Person2 p2;

GO -- Run the prvious command and begins new batch

/\*

Person has 10 rows

Person2 has 10 rows

Compare Person and Person2, there are 5 rows are the same.

-->

UNION removes duplicate rows,

UNION ALL does not.

-->

UNION

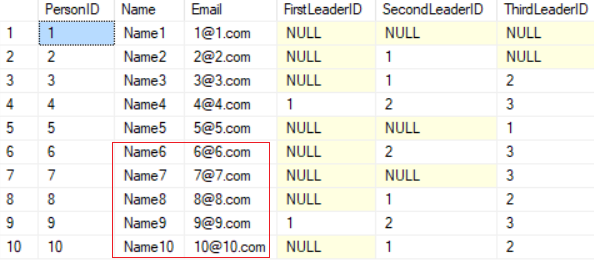
-->

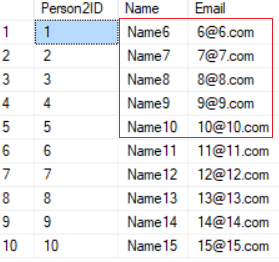
(10 rows from Person

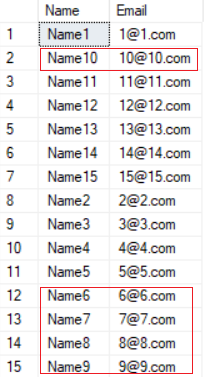
+ 10 rows from Person2

- 5 duplicate rows) = 15 rows

\*/







--=============================================================================

--T005\_08\_03

--UNION(ALL)...OrderBy

SELECT  \*

FROM    dbo.Employee;

SELECT  \*

FROM    dbo.Employee

WHERE   DepartmentID = 1

--ORDER BY FirstName;   --ERROR

UNION ALL

SELECT  \*

FROM    dbo.Employee

WHERE   DepartmentID = 2

--ORDER BY FirstName;   --ERROR

UNION ALL

SELECT  \*

FROM    dbo.Employee

WHERE   DepartmentID = 3

ORDER BY DepartmentID;

GO -- Run the prvious command and begins new batch

/\*

1.

DepartmentID = 1  has 3 rows

DepartmentID = 2  has 3 rows

DepartmentID = 3  has 2 rows

-->

UNION removes duplicate rows,

UNION ALL does not.

-->

UNION ALL

-->

(3 rows from DepartmentID = 1

+ 3 rows from DepartmentID = 2

+ 2 rows from DepartmentID = 3) = 8 rows

2.

ORDER BY caluse can only be used on the last SELECT statement.

ORDER BY caluse is on any other SELECT statement will cause Syntax Error.

UNION combines rows from 2 or more tables/Search Results.

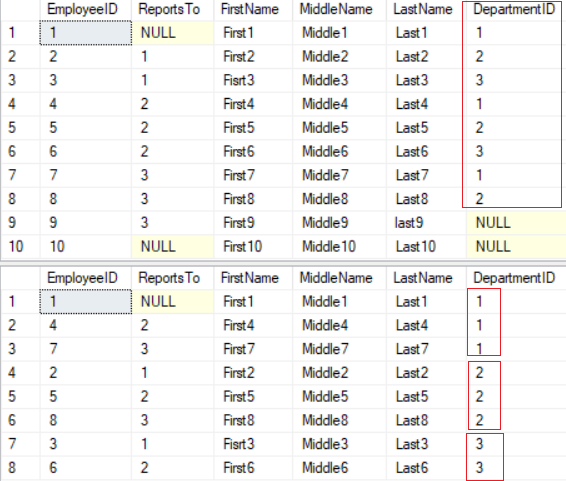
Thus, ORDER BY caluse can only be used after all the results is combined.

3.

UNION combines rows from 2 or more tables/Search Results.

JOINS combine columns from 2 or more tables.

\*/



--=============================================================================

--T005\_08\_04

--Different between UNION(ALL) and JOINS

SELECT  \*

FROM    dbo.Employee;

SELECT  \*

FROM    dbo.Department;

SELECT  e.EmployeeID ,

        ( e.FirstName + ' ' + e.MiddleName + ' ' + e.LastName ) AS [Name] ,

        d.DepartmentID ,

        d.DepartmentName

FROM    dbo.Employee e

        INNER JOIN dbo.Department d ON e.DepartmentID = d.DepartmentID

ORDER BY d.DepartmentID;

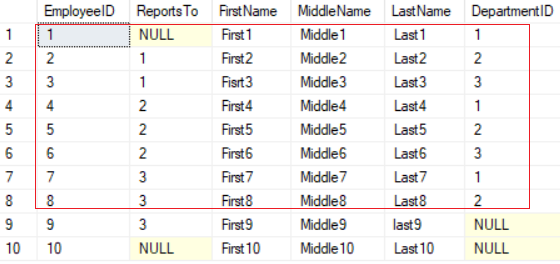
GO -- Run the prvious command and begins new batch

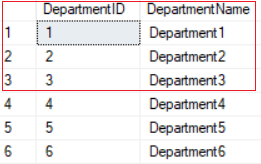
/\*

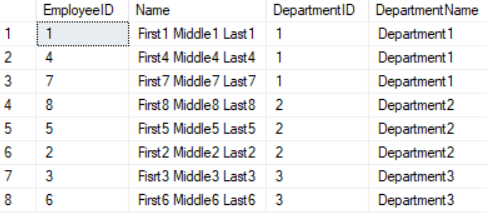
UNION (ALL) combines rows from 2 or more tables/Search Results.

JOINS combine columns from 2 or more tables.

\*/







--=============================================================================

--T005\_09\_Clean up

--=============================================================================

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Employee' ) )

    BEGIN

        TRUNCATE TABLE Employee;

        DROP TABLE Employee;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Department' ) )

    BEGIN

        TRUNCATE TABLE Department;

        DROP TABLE Department;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'CarModel' ) )

    BEGIN

        TRUNCATE TABLE CarModel;

        DROP TABLE CarModel;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'CarColor' ) )

    BEGIN

        TRUNCATE TABLE CarColor;

        DROP TABLE CarColor;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Player' ) )

    BEGIN

        TRUNCATE TABLE Player;

        DROP TABLE Player;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Person' ) )

    BEGIN

        TRUNCATE TABLE Person;

        DROP TABLE Person;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Person2' ) )

    BEGIN

        TRUNCATE TABLE Person2;

        DROP TABLE Person2;

    END;

GO -- Run the previous command and begins new batch