CMC CORPORATION
Towards the digital future

Advanced DML Statements

Agenda

- JOINs in SQL Server
- Subquery
- Common Table Expressions(CTE)
- Ranking functions
- SQL code practices

Overview

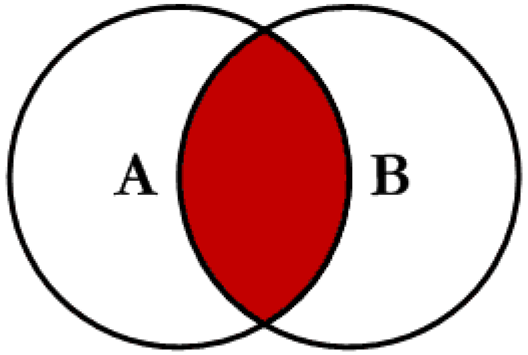
- **JOINs:** Retrieve data from two or more tables based on logical relationships between the tables:
  - ✓ *Inner Join*
  - ✓ *Outer Join*
  - ✓ *Cross Join*
  - ✓ *Self Join*

- **Subquery:** A query that is nested inside a SELECT, INSERT, UPDATE, or DELETE statement, or inside another sub-query

- **Ranking functions:**
  - ✓ *Row_Number*
  - ✓ *Rank*
  - ✓ *Dense_Rank*
  - ✓ *Ntitle*

CMC CORPORATION
Towards the digital future

JOINs in SQL Server
INNER JOIN



- Return all of the records in the left table (table A) that have a matching record in the right table (table B)
  - ✓ *Eliminate the rows that do not match with a row from the other table*

- **Syntax**

```
 SELECT col_names
FROM Table_A  A
INNER JOIN Table_B  B
 ON A.Col1 = B.Col1
```

CMC CORPORATION
Towards the digital future

JOINs in SQL Server
INNER JOIN Demo

- **Demo**

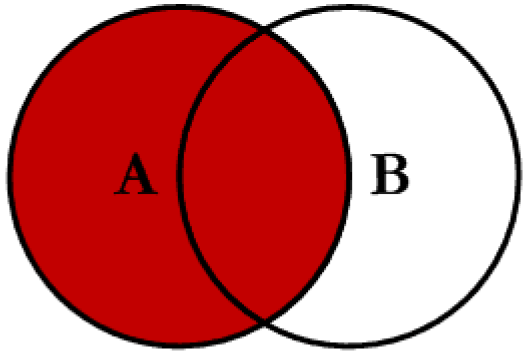
JOINs in SQL Server
OUTER JOIN

- **Outer Join:** Return all rows from at least one of the tables mentioned in the FROM clause, as long as those rows meet any WHERE or HAVING search conditions:

  - ✓ LEFT OUTER JOIN (or LEFT JOIN)
  - ✓ RIGHT OUTER JOIN (or RIGHT JOIN)
  - ✓ FULL OUTER JOIN (or FULL JOIN)

CMC CORPORATION
Towards the digital future

JOINs in SQL Server
LEFT OUTER JOIN



- Return all of the records in the left table (table A) regardless if any of those records have a match in the right table (table B)

  ✓ In the results where there is no matching condition, the row contains NULL values for the right table’s columns.

- **Syntax**

```
  SELECT col_names
FROM Table_A  A
LEFT JOIN Table_B  B
  ON A.Col1 = B.Col1
```

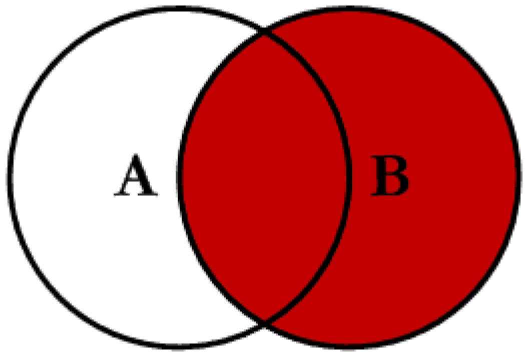
CMC CORPORATION
Towards the digital future

JOINs in SQL Server
LEFT OUTER JOIN Demo

- **Demo**

CMC CORPORATION
Towards the digital future

JOINs in SQL Server
RIGHT OUTER JOIN



- Return all of the records in the right table (table B) regardless if any of those records have a match in the left table (table A)

✓ *In the results where there is no matching condition, the row contains NULL values for the left table's columns.*

- **Syntax**

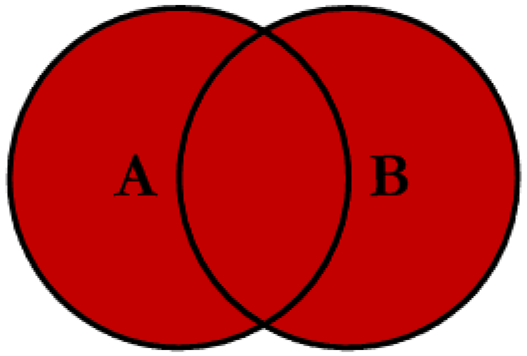
```
  SELECT col_names
FROM Table_A  A
RIGHT JOIN Table_B  B
  ON A.Col1 = B.Col1
```

CMC CORPORATION
Towards the digital future

JOINs in SQL Server
RIGHT OUTER JOIN Demo

- **Demo**

JOINs in SQL Server FULL OUTER JOIN



- Return all of the records from both tables, joining records from the left table (table A) that match records from the right table (table B)

- **Syntax**

```
   SELECT col_names
FROM Table_A  A
FULL JOIN Table_B  B
   ON A.Col1 = B.Col1
```

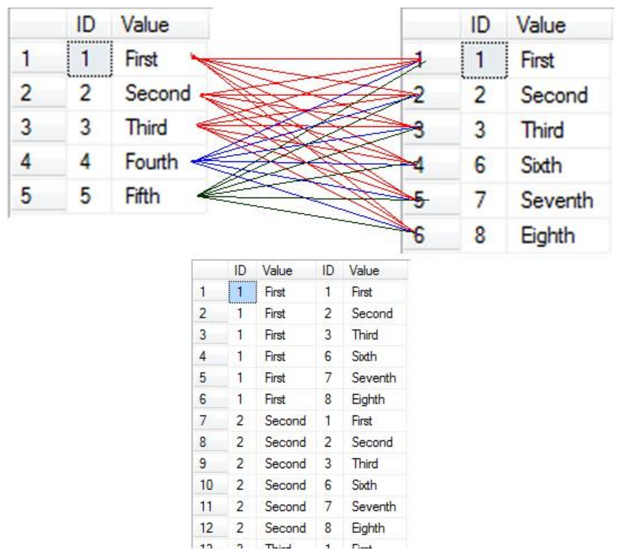
CMC CORPORATION
Towards the digital future

JOINs in SQL Server
FULL OUTER JOIN Demo

- **Demo**

CMC CORPORATION
Towards the digital future

JOINs in SQL Server
CROSS JOIN



	ID	Value
1	1	First
2	2	Second
3	3	Third
4	4	Fourth
5	5	Fifth

	ID	Value
1	1	First
2	2	Second
3	3	Third
4	6	Sixth
5	7	Seventh
6	8	Eighth

	ID	Value	ID	Value
1	1	First	1	First
2	1	First	2	Second
3	1	First	3	Third
4	1	First	6	Sixth
5	1	First	7	Seventh
6	1	First	8	Eighth
7	2	Second	1	First
8	2	Second	2	Second
9	2	Second	3	Third
10	2	Second	6	Sixth
11	2	Second	7	Seventh
12	2	Second	8	Eighth

- Return records that are multiplication of record number from both the tables
  - ✓ Does not need any condition to join

- Syntax:

```
SELECT col_names
FROM Table_A  A
CROSS JOIN Table_B  B
```

CMC CORPORATION
Towards the digital future

JOINs in SQL Server
CROSS JOIN Demo

- **Demo**

JOINs in SQL Server
Self JOIN

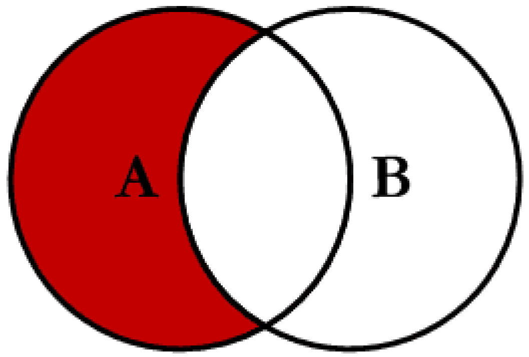
- **A SELF JOIN is a join of a table to itself. In SELF JOIN, we can use:**
  - INNER JOIN
  - OUTER JOIN
  - CROSS JOIN

JOINs in SQL Server
SELF JOIN Demo

- **Demo**

CMC CORPORATION
Towards the digital future

JOINs in SQL Server
LEFT Excluding JOIN



- Return all of the records in the left table (table A) that do not match any records in the right table (table B)

- **Syntax**

```
  SELECT col_names
FROM Table_A  A
LEFT JOIN Table_B  B
  ON A.Col1 = B.Col1
  WHERE B.Col1 IS NULL
```

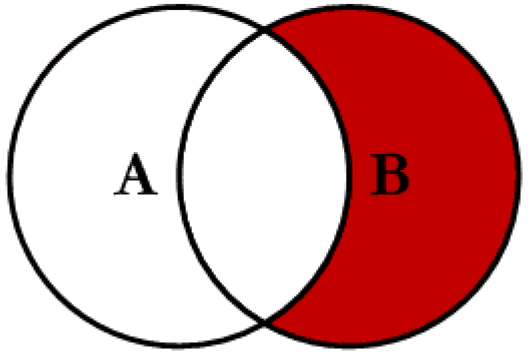
CMC CORPORATION
Towards the digital future

JOINs in SQL Server
LEFT Excluding JOIN Demo

- **Demo**

CMC CORPORATION
Towards the digital future

JOINs in SQL Server RIGHT Excluding JOIN



- Returns records in the right table (table B) that do not match any records in the left table (table A)

- **Syntax**

```
  SELECT col_names
FROM Table_A  A
RIGHT JOIN Table_B  B
  ON A.Col1 = B.Col1
  WHERE A.Col1 IS NULL
```

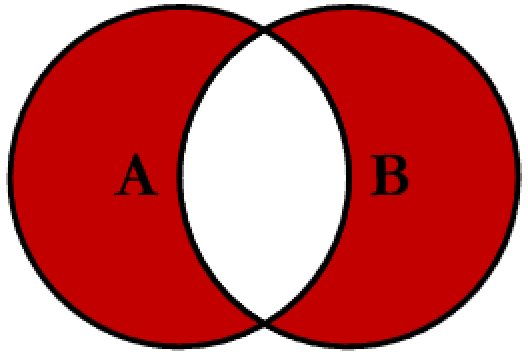
CMC CORPORATION
Towards the digital future

JOINs in SQL Server
RIGHT Excluding JOIN Demo

- **Demo**

CMC CORPORATION
Towards the digital future

JOINs in SQL Server
OUTER JOIN EXCLUDING JOIN



- Return all of the records in the left table (table A) and all of the records in the right table (table B) that do not match

- **Syntax**

```
  SELECT col_names
FROM Table_A  A
RIGHT JOIN Table_B  B
  ON A.Col1 = B.Col1
  WHERE A.Col1 IS NULL OR B.Col1 IS NULL
```

CMC CORPORATION
Towards the digital future

JOINs in SQL Server
OUTER JOIN EXCLUDING JOIN Demo

- **Demo**

CMC CORPORATION
Towards the digital future

JOINs in SQL Server
Joining Three or More Tables

- Due to FROM clauses can contain multiple join specifications so this allows many tables to be joined for a single query

- **Example**

```
  SELECT col_names
FROM Table_A  A
JOIN Table_B  B
  ON A.Col1 = B.Col1
  LEFT JOIN Table_C C
  ON B.Col2 = C.Col2
  ….
```

JOINs in SQL Server
Joining Three or More Tables Demo

- **Demo**

Subquery

- **Subquery:** Is a query that is nested inside a SELECT, INSERT, UPDATE, or DELETE statement, or inside another sub-query

✓ *Inner query is independent of outer query.*

✓ *Inner query is executed first and the results are stored.*

✓ *Outer query then runs on the stored results.*

Subquery
Subquery Types

- **We focus on some types of Subquery:**
  - **Subqueries with Aliases**
    - Many statements in which the subquery and the outer query refer to the same table
  - **Subqueries with IN / NOT IN**
    - The result of a subquery introduced with IN (or with NOT IN) is a list of zero or more values. After the subquery returns results, the outer query makes use of them
  - **Subqueries in UPDATE, DELETE, INSERT, SELECT**
  - **Subqueries with EXISTS / NOT EXISTS**
    - The subquery functions as an existence test.

Subquery
Subquery Types Demo

- **Demo**
  - ✓ Subqueries with Aliases
  - ✓ Subqueries with IN / NOT IN
  - ✓ Subqueries in UPDATE, DELETE, INSERT, SELECT
  - ✓ Subqueries with EXISTS / NOT EXISTS

Common Table Expressions

- A CTE can be thought of as a temporary result set that is defined within the execution scope of a single SELECT, INSERT, UPDATE, DELETE. It can be used:
  - ✓ *Create a recursive query*
  - ✓ *As a temporary table*

- **Syntax**

```
; WITH CTE_Name [ col_names]
AS
(
 CTE_query_definition
)
```

CMC CORPORATION
Towards the digital future

Common Table Expressions Recursive

- **Recursive Queries Using Common Table Expressions**

  - **Syntax:**

```
WITH cte_name ( col_names)
AS
(
CTE_query_definition  -- Anchor member is defined.
UNION ALL
CTE_query_definition  -- Recursive member is defined referencing cte_name.
)
-- Statement using the CTE
SELECT *
FROM cte_name
```

Common Table Expressions
Common Table Expressions Demo

- **Demo**

CMC CORPORATION
Towards the digital future

Ranking functions

- **Ranking functions:** Ranking functions provides the ability to rank each row of data.

  - **Row_Number**: Returns the sequential number of a row within a partition of a result set
  - **Rank**: Returns the rank of each row within the partition of a result set
  - **Dense_Rank**: Returns the rank of rows within the partition of a result set, without any gaps in the ranking
  - **Ntitle**: Distributes the rows in an ordered partition into a specified number of groups

Ranking functions Demo

- **Demo**
  - Row_Number
  - Rank
  - Dense_Rank
  - NTitle

SQL Code Practice

- **Explicitly Name Columns in SELECT Statements**
  - ✓ Improve performance.
  - ✓ Prevent potential failures related to some database schema change in the future.

- **For example, using:**

```
SELECT EmployeeID, FirstName, LastName FROM dbo.Employee
```

Instead of:

```
SELECT * FROM dbo.Employee
```

SQL Code Practice

- **Explicitly Name Columns in INSERT Statements**
  - Prevent potential failures related to some database schema change in the future.
  - Prevent error with identity column

- **For example, using:**

```
INSERT dbo.Employee (FirstName, LastName, NationalIDNumber, ManagerID, Title,
  BirthDate, MaritalStatus, Gender)
VALUES ('Bill', 'Gates', '123456', NULL, 'CEO', '1959-01-01', 'M' , 'M')
```

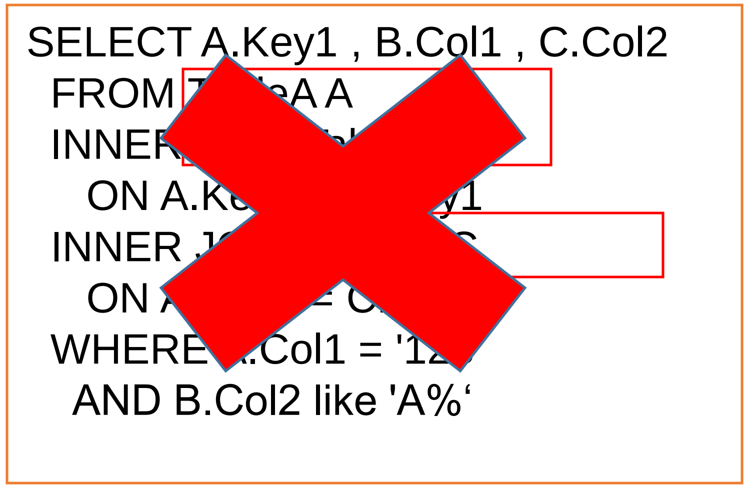
Instead of :

```
INSERT dbo.Employee
VALUES ('Bill', 'Gates', '123456', NULL, 'CEO', '1959-01-01', 'M' , 'M')
```

SQL Code Practice

- **Always specific schema for tables in query.**
  - Prevent potential failures related to some database schema change or permission change on schema in the future.

```
SELECT A.Key1 , B.Col1 , C.Col2
  FROM dbo.TableA A
  INNER JOIN dbo.TableB  B
    ON A.Key1 = B.Key1
  INNER JOIN dbo.TableC C
    ON A.Key1 = C.Key1
  WHERE A.Col1 = '123'
   AND B.Col2 like 'A%‘
```



```
SELECT A.Key1 , B.Col1 , C.Col2
  FROM T      eA A
  INNER
    ON A.Ke           y1
  INNER J            C
    ON A           = C
  WHERE  .Col1 = '12
   AND B.Col2 like 'A%‘
```

SQL Code Practice

- **Always provides alias for tables in query.**
  - ✓ ake query more clearer and easier to read.

```
SELECT A.Key1 , B.Col1 , C.Col2
  FROM dbo.TableA A
  INNER JOIN dbo.TableB  B
     ON A.Key1 = B.Key1
  INNER JOIN dbo.TableC C
     ON A.Key1 = C.Key1
  WHERE A.Col1 = '123'
    AND B.Col2 like 'A%‘
```

```
SELECT TableA.Key1 , TableB.Col1 ,
TableC.Col2
  FROM dbo.[illegible]bleA
  INNER JO[illegible]
     ON TableA.[illegible]TableB.Key1
  INNER JOI[illegible]C
     ON Tab[illegible]ey1[illegible]leC.Key1
  WHERE TableA.Col1 = '123'
    AND TableB.Col2 like 'A%‘
```

SQL Code Practice

- **Avoid SQL Server functions in the WHERE clause**
  - ✓ Improve performance.

```
SELECT EmailAddress
FROM person.contact
WHERE EmailAddress like 'As%'
```

```
SELEC[illegible]EmailAddr[illegible]
FROM per[illegible]act
WHER[illegible]t(Emai[illegible]ess,2) = 'As'
```

SQL Code Practice

- **Only use DISTINCT if necessary**
- **Only use UNION if necessary, in other case use UNION ALL**

CMC CORPORATION
Towards the digital future

Q & A

- How many JOINs in SQL Server?
- Is there any different if we put condition at ON condition and WHERE condition of INNER JOIN?
- Is there any different if we put condition at ON condition and WHERE condition of LEFT JOIN?
- When should we use subquery instead of JOIN?