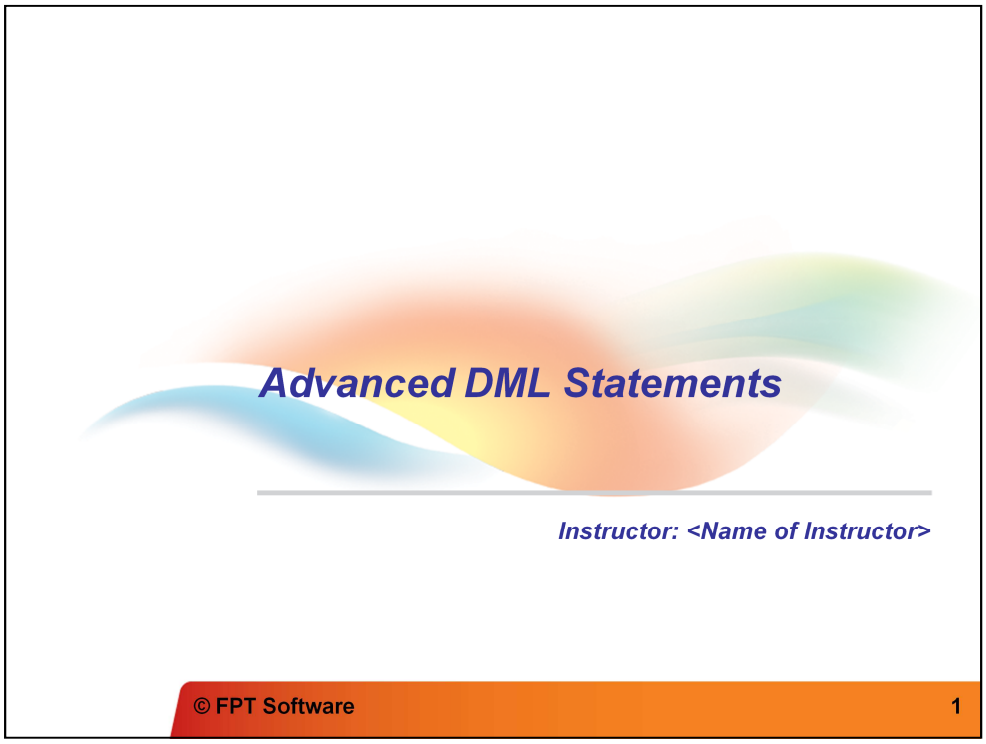


Latest updated by: HanhTT1

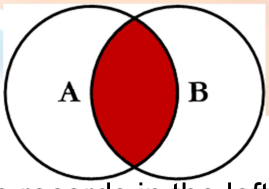
Agenda

- 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

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**

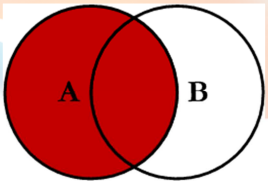
```
SELECT col_names
FROM Table_A  A
      INNER JOIN Table_B  B ON A.Col1 = B.Col1
```

See “INNER JOIN” in Day1_Demo.docx

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)

JOINs in SQL Server
LEFT OUTER JOIN

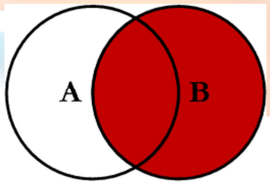


- Return all of the records in the left table (table A) regardless if any of those records have a match in the right table (table B)
  - In the results where there is no matching condition, the row contains NULL values for the right table's columns.
- **Syntax**

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

See “LEFT JOIN” in Day1_Demo.docx

JOINs in SQL Server
RIGHT OUTER JOIN

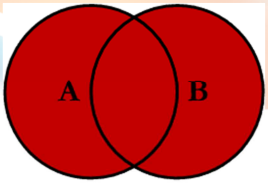


- Return all of the records in the right table (table B) regardless if any of those records have a match in the left table (table A)
  - In the results where there is no matching condition, the row contains NULL values for the left table's columns.
- **Syntax**

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

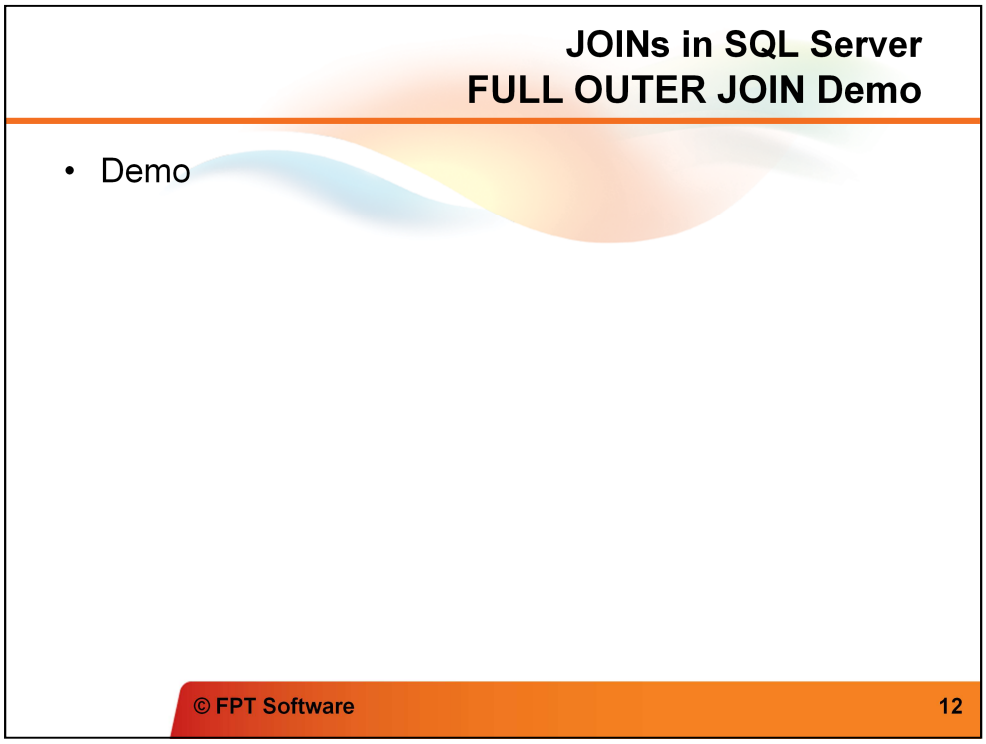
See “RIGHT JOIN” in Day1_Demo.docx

JOINs in SQL Server
FULL OUTER JOIN



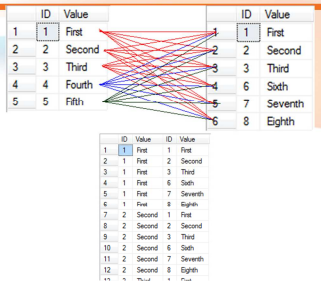
- Return all of the records from both tables, joining records from the left table (table A) that match records from the right table (table B)
- **Syntax**

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



See “FULL JOIN” in Day1_Demo.docx

JOINs in SQL Server
CROSS JOIN



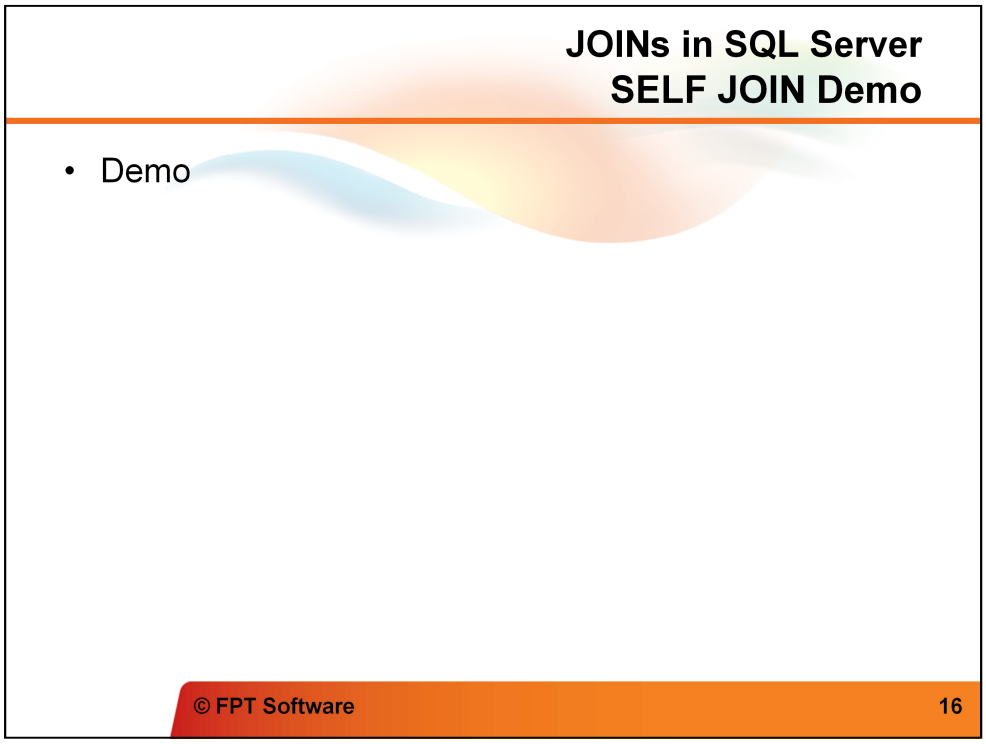
- Return records that are multiplication of record number from both the tables
  - Does not need any condition to join
- Syntax:

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

See “CROSS JOIN” in Day1_Demo.docx

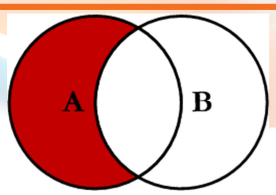
JOINs in SQL Server
Self JOIN

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



See “Seft Join” in Day1_Demo.docx

JOINs in SQL Server
LEFT Excluding JOIN

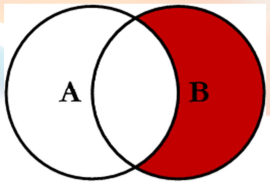


- Return all of the records in the left table (table A) that do not match any records in the right table (table B)
- **Syntax**

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

See “LEFT Excluding JOIN” in Day1_Demo.docx

JOINs in SQL Server
RIGHT Excluding JOIN

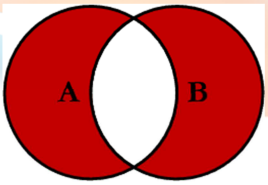


- Returns records in the right table (table B) that do not match any records in the left table (table A)
- **Syntax**

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

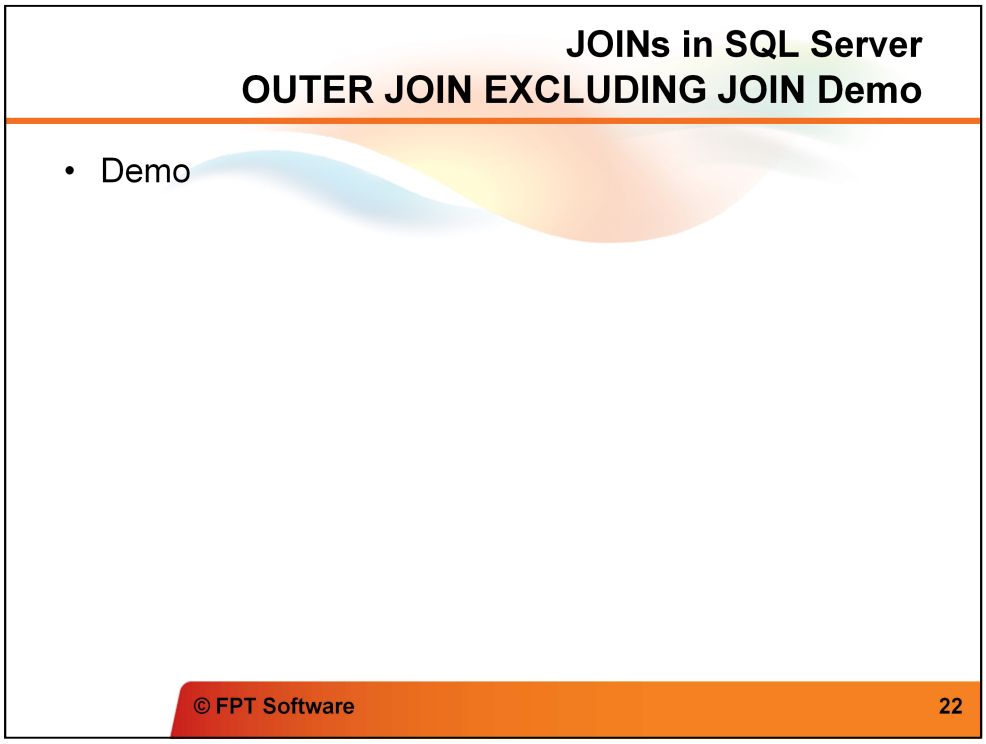
See Day1_Demo.docx

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
```



See “OUTER JOIN EXCLUDING JOIN” in Day1_Demo.docx

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)
....
```

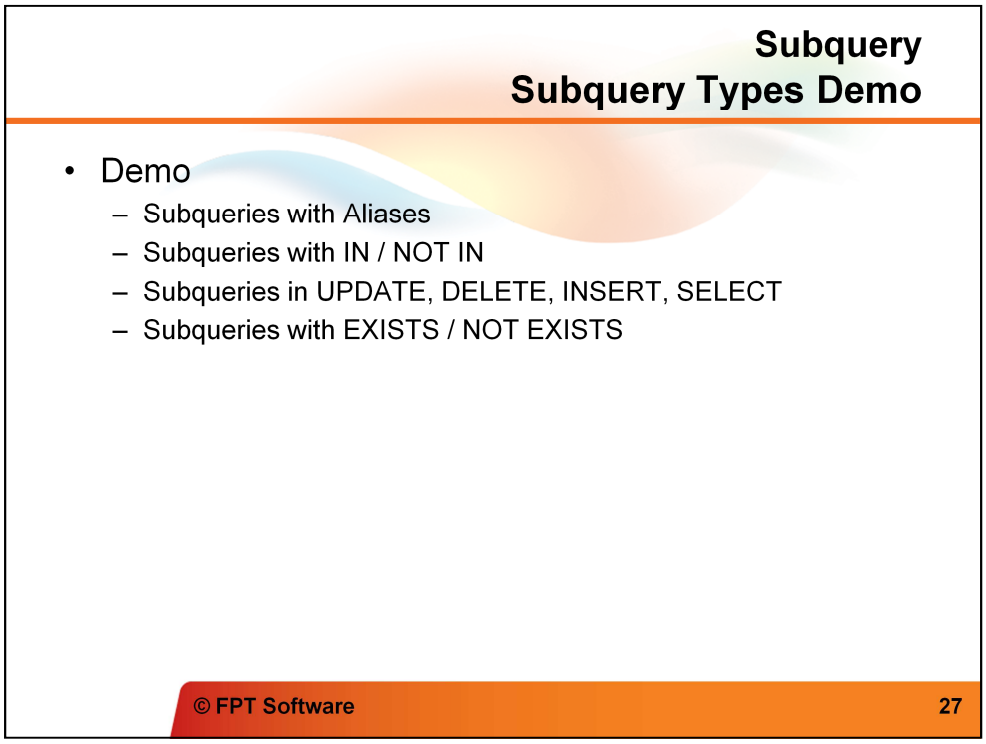
See Day1_Demo.docx

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.



See “Subquery” in Day1_Demo.docx

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.

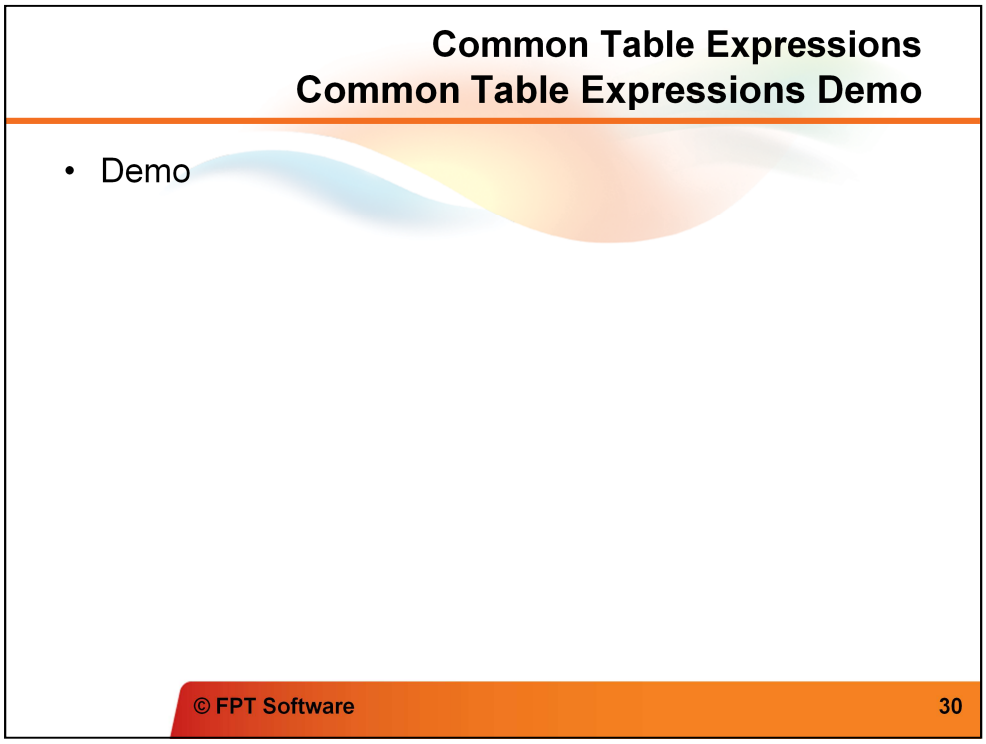
- Syntax

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

Common Table Expressions Recursive

- **Recursive Queries Using Common Table Expressions**
  - **Syntax:**

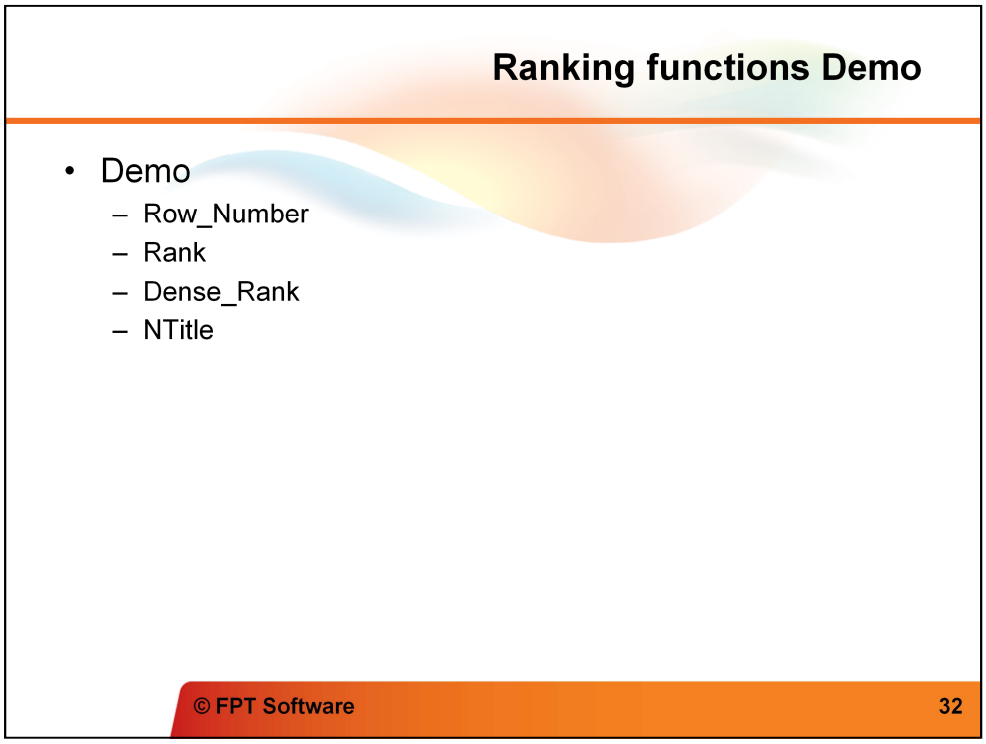
```
WITH cte_name ( col_names)
AS
(
CTE_query_definition  -- Anchor member is defined.
UNION ALL
CTE_query_definition  -- Recursive member is defined
   referencing cte_name.
)
-- Statement using the CTE
SELECT *
FROM cte_name
```



See “Common Table Expressions” in Day1_Demo.docx

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



See “Ranking functions” in Day1_Demo.docx

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')
```

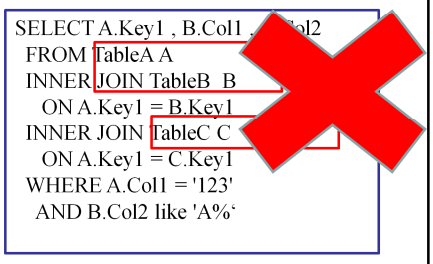
Instead of :

```
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%'
```



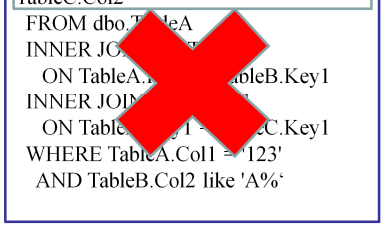
```
SELECT A.Key1 , B.Col1 , C.Col2
  FROM TableA A
  INNER JOIN TableB  B
    ON A.Key1 = B.Key1
  INNER JOIN TableC C
    ON A.Key1 = C.Key1
  WHERE A.Col1 = '123'
    AND B.Col2 like 'A%'
```


Cập nhật lại slide

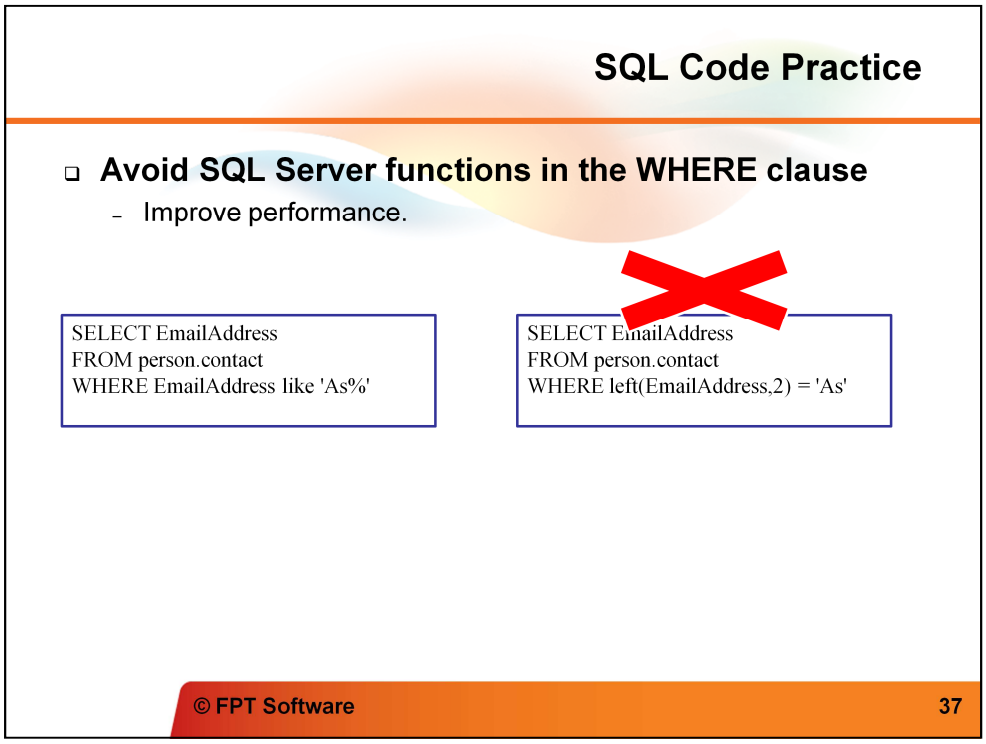
SQL Code Practice

- **Always provides alias for tables in query.**
  - Make 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%'
```



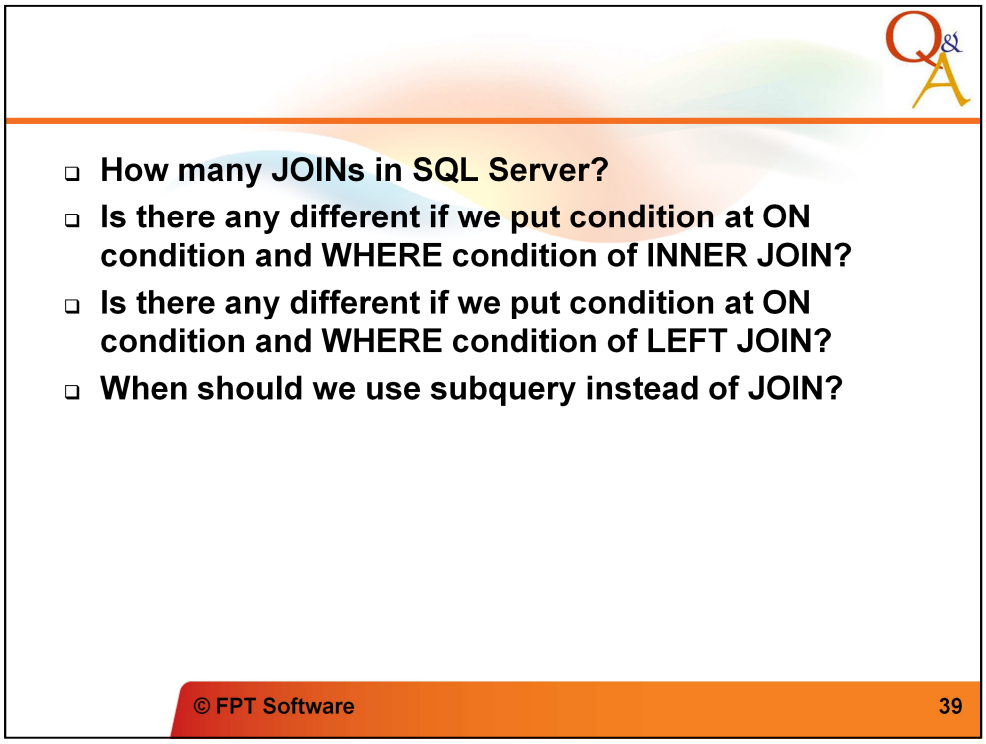
```
SELECT TableA.Key1 , TableB.Col1 ,
TableC.Col2
  FROM dbo.TableA
  INNER JOIN dbo.TableB
    ON TableA.Key1 = TableB.Key1
  INNER JOIN dbo.TableC
    ON TableA.Key1 = TableC.Key1
  WHERE TableA.Col1 = '123'
    AND TableB.Col2 like 'A%'
```



See more detail in the “SQL Code Practice” in Day1_Demo.docx

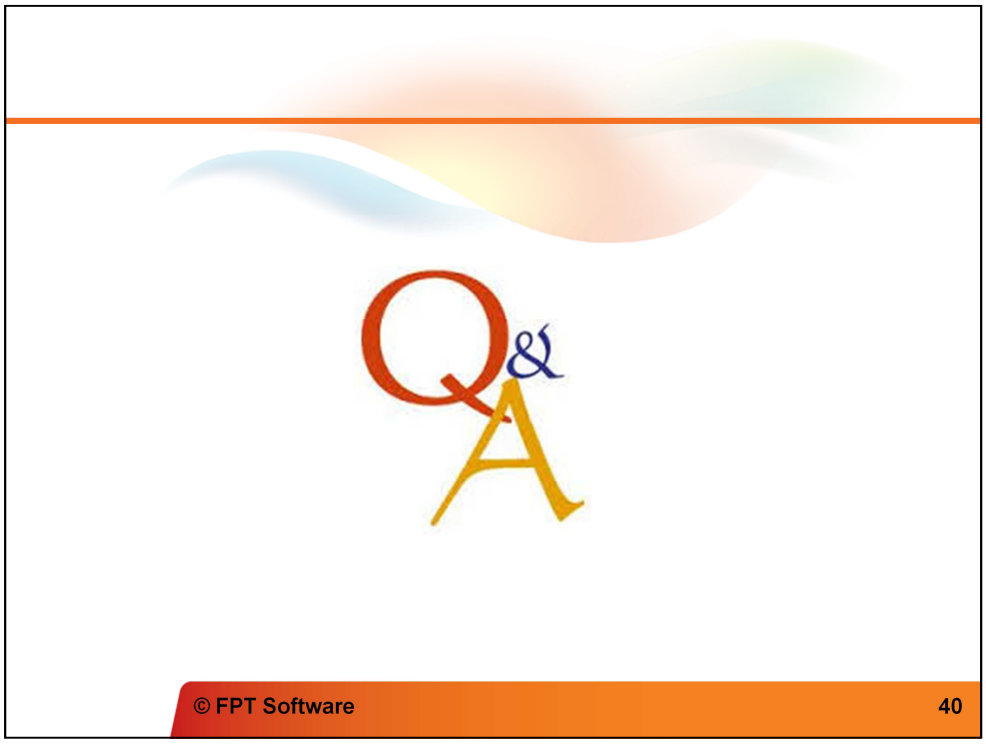
SQL Code Practice

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



When should we use subquery instead of JOIN?

Đưa ra ví dụ dùng cả hai được -> Subquery và Join


Q&A
© FPT Software
40