Video-19

Topics to cover:

- Recursive Common Table Expression(CTE) in SQL

# Recursive CTE

This is a CTE that references itself. In other words, the CTE query definition refers back to the CTE name, creating a loop that ends when a certain condition is met.

A recursive CTE has three elements:

- **Non-recursive term**: It's a CTE query definition that forms the base result set of the CTE structure.
- **Recursive term**: One or more CTE query definitions joined with non-recursive term using UNION or UNION ALL operator.
- **Termination check**: the recursion stops when no rows are returned from the previous iteration.

Syntax

```
WITH RECURSIVE cte_name AS (
        CTE_query_definition -- non recursive term(base query/anchor member)
        UNION ALL
        recursive_query_definition -- recursive term (recursive query / recursive member)
)
SELECT * FROM cte_name;
```

# Recursive CTE

# Example - 1

```
WITH RECURSIVE cte_count AS (
	SELECT 1 AS n -- non recursive term(base query/anchor member)
	UNION ALL
	SELECT n+1 FROM cte_count -- recursive term (recursive query / recursive member)
	Where n<5 ) -- Termination check

SELECT * FROM cte_count;
```

Recursive CTE

Example - 2 : Recursive Date Generator

```
CREATE TABLE CalendarDates (
    DateValue DATE PRIMARY KEY
);

WITH RECURSIVE DatesCTE AS (
    SELECT DATE('2026-01-01') AS DateValue -- Anchor: start with the first day of the month

    UNION ALL

    SELECT DateValue + INTERVAL 1 DAY -- Recursive: add one day until the end of the month
    FROM DatesCTE
    WHERE DateValue < '2026-01-05'
)
SELECT * FROM DatesCTE;
```

# Recursive CTE

# Example - 3 Finding Employees Hierarchy

```
CREATE TABLE employees (
        emp_id int PRIMARY KEY,
   emp_name VARCHAR(50) NOT NULL,
        manager_id INT
);

INSERT INTO employees (emp_id, emp_name, manager_id) VALUES
(1, 'Alice', NULL),
(2, 'Bob', 1),
(3, 'Carol', 2),
(4, 'David', 6),
(5, 'Eva', 4),
(6, 'Tom', 1),
(7, 'Frank', 5);

SELECT * FROM employees;
```

# Recursive CTE

```
WITH RECURSIVE EmployeeHierarchy AS (
    -- Anchor: start with root manager(s)
    SELECT emp_id, emp_name, manager_id
    FROM employees
    WHERE emp_id = 7

    UNION ALL

    -- Recursive: find direct reports
    SELECT employees.emp_id, employees.emp_name, employees.manager_id
    FROM employees
    JOIN EmployeeHierarchy
          ON employees.emp_id = EmployeeHierarchy.manager_id
)
SELECT * FROM EmployeeHierarchy;
```