

# SQL : Common Table Expression(CTE)

---

# What is CTE? : Common Table Expressions

---

- It is a **temporary result set** that can be referenced within the context of a **SELECT, INSERT, UPDATE, or DELETE** statement in SQL.

```
SELECT EmployeeID, FirstName, LastName
FROM Employees
WHERE DepartmentID = (SELECT
DepartmentID FROM Departments WHERE
DepartmentName = 'Sales');
```

----

```
WITH SalesDepartment AS (
    SELECT DepartmentID
    FROM Departments
    WHERE DepartmentName = 'Sales'
)
SELECT EmployeeID, FirstName, LastName
FROM Employees
WHERE DepartmentID = (SELECT
DepartmentID FROM SalesDepartment);
```

# Syntax of CTEs

---

```
WITH MyCTE AS (  
    SELECT Column1, Column2  
    FROM MyTable  
)
```

```
SELECT * FROM MyCTE;
```

# Recursive CTEs

---

```
WITH EmployeeHierarchy AS (  
    SELECT EmployeeID, ManagerID  
    FROM EmployeeTable  
    WHERE ManagerID IS NULL -- Root level employees  
  
    UNION ALL  
    SELECT e.EmployeeID, e.ManagerID  
    FROM EmployeeTable e  
    JOIN EmployeeHierarchy eh ON e.ManagerID = eh.EmployeeID  
)  
  
SELECT * FROM EmployeeHierarchy;
```

# Why CTEs are useful?

---

- Break down **complex** Queries into **Smaller**, more manageable parts
- Improve **readability** & **maintainability** of SQL code.
- Useful in **recursive** Queries, handling **hierarchical** data & performing **multi-step** data transformations.
- Reduce code **redundancy**.
- Easier to **write** & **debug** the complex queries.

# Example

---

- Demo

# CTEs in Stored Procedures

---

-- Create a stored procedure that uses a CTE

CREATE PROCEDURE MyStoredProcedure

AS

BEGIN

-- Define the CTE

WITH MyCTE AS (

SELECT Column1, Column2

FROM MyTable

WHERE SomeCondition

)

-- Use the CTE within the stored procedure

SELECT \* FROM MyCTE;

END;

# CTEs in INSERT statement

-- Insert records into the "Orders" table based on data from the "Customers" table

```
WITH OrderData AS (  
    SELECT CustomerID  
    FROM Customers  
    WHERE Country = 'USA'  
)  
INSERT INTO Orders (CustomerID, OrderID, OrderDate)  
SELECT CustomerID, 3, GETDATE() AS OrderDate FROM OrderData;
```



# CTEs in UPDATE statement

-- Update customer names in the "Customers" table based on conditions and data from the "Orders" table

```
WITH UpdatedCustomers AS (  
    SELECT c.CustomerID, o.OrderDate  
    FROM Customers c  
    JOIN Orders o ON c.CustomerID = o.CustomerID  
    WHERE o.OrderDate >= '2023-01-01'  
)  
UPDATE Customers  
SET CustomerName = 'Valued Customer'  
WHERE CustomerID IN (SELECT CustomerID FROM UpdatedCustomers);
```

# CTEs in DELETE statement

--Delete customers from the "Customers" table based on conditions and data from the "Orders" table

```
WITH CustomersToDelete AS (  
    SELECT c.CustomerID  
    FROM Customers c  
    LEFT JOIN Orders o ON c.CustomerID = o.CustomerID  
    WHERE o.OrderDate IS NULL  
)  
DELETE FROM Customers  
WHERE CustomerID IN (SELECT CustomerID FROM CustomersToDelete);
```

# Best Practices for Writing CTEs

- Use Descriptive CTE Names
- Format CTE Queries for readability
- Add comments for explanation
- Avoid Recursive CTE loops
- Use CTEs for Code reusability
- Optimize with indexes
- Test & Debug thoroughly
- Document your CTEs