# Common Table Expressions Using WITH Part 1 Notes

Team Treehouse

June 8, 2020

## 1 What is a Common Table Expression?

- Works like function in programming
- Makes queires easier to read
- Organizes queires into **reusable** modules
- Better matches to how you think about data analysis
- Uses WITH

**Example:**

```
    WITH product_details AS (
        SELECT ProductName, CategoryName, UnitPrice, UnitInStock
        FROM Products
        JOIN Categories ON PRODUCTS.CategoryId = Categories.id
        WHERE Products.Discountinued = 0
    )

    SELECT * FROM product_details // <- Noticed it's used like a
    function
    ORDER BY CategoryName, ProductName

```

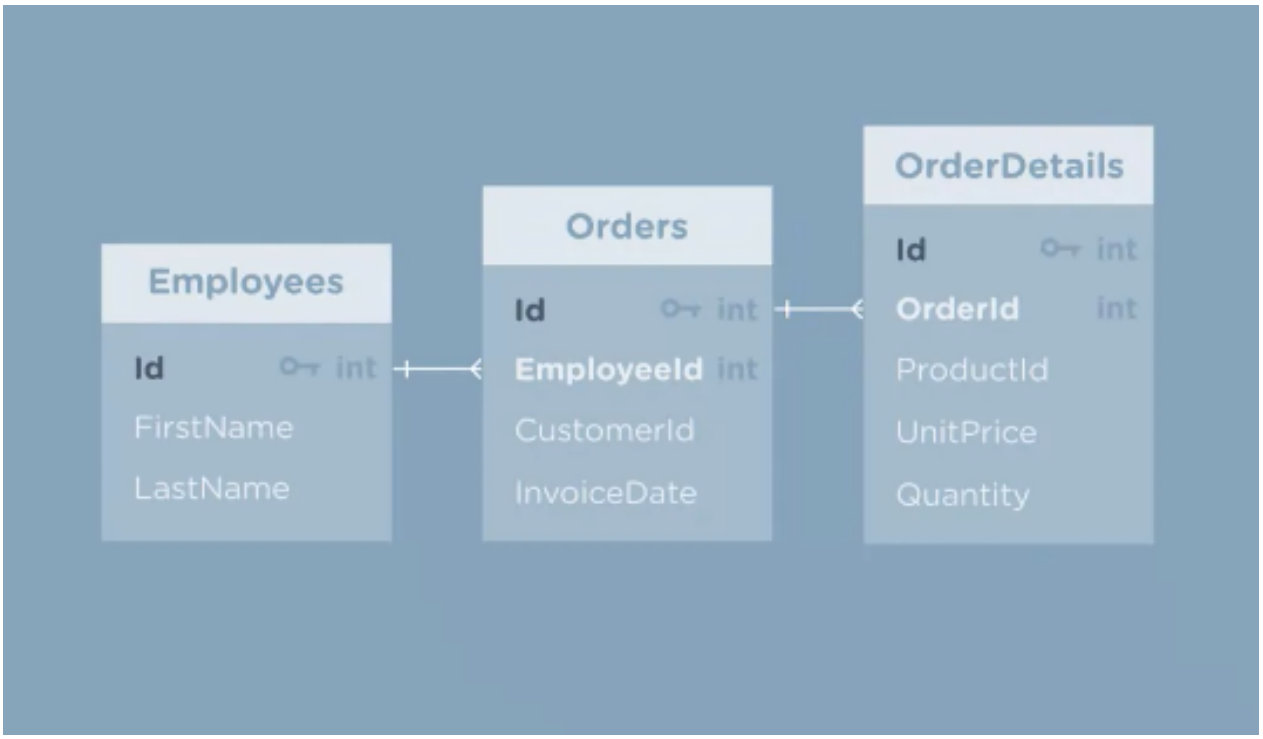
# 2 Convert a Subquery to a CTE

- To declare multiple CTES, WITH is required only once

**Example:**

```
\\ ======= BEFORE CTE =======

SELECT all_orders.EmployeeID, Employees.LastName, all_orders.order_count AS total_order_count, late_orders.order_count AS late_order_count
FROM (
    SELECT EmployeeID, COUNT(*) AS order_count
    FROM Orders
    GROUP BY EmployeeID
) all_orders
JOIN (
  SELECT EmployeeID, COUNT(*) AS order_count
  FROM Orders
  WHERE RequiredDate <= ShippedDate
  GROUP BY EmployeeID
) late_orders
ON all_orders.EmployeeID = late_orders.employeeID
JOIN Employees
ON all_orders.EmployeeId = Employees.Id


\\ ======= AFTER CTE ========

SELECT EmployeeID, COUNT(*) AS order_count
    FROM Orders
    GROUP BY EmployeeID
),
late_orders AS (
    SELECT EmployeeID, COUNT(*) AS order_count
    FROM Orders
    WHERE RequiredDate <= ShippedDate
    GROUP BY EmployeeID
)
SELECT Employees.ID, LastName, all_orders.order_count AS total_order_count, late_orders.order_count AS late_order_count
FROM Employees
JOIN all_orders ON Employees.ID = all_orders.EmployeeID
JOIN late_orders ON Employees.ID = late_orders.EmployeeID

```

# 3 Using Multiple CTEs in a Query




- can only reference earlier WITH expression
- cannot reference latter WITH expressions

**Example:**

```
    WITH
    all_sales AS (
        SELECT Orders.Id AS OrderId, Orders.EmployeeId,
        SUM(OrderDetails.UnitPrice * OrderDetails.Quantity) AS
   invoice_total
        FROM Orders
        JOIN OrderDetails ON Orders.id = OrderDetails.OrderId
        GROUP BY Orders.ID
    ),
    revenue_by_employee AS (
        SELECT EmployeeId, SUM(invoice_total) AS total_revenue
        FROM all_sales //<- From Earlier WITH
        GROUP BY EmployeeID
    ),
    sales_by_employee AS (
        SELECT EmployeeId, COUNT(*) AS sales_count
        FROM all_sales //<- From Earlier WITH
        GROUP BY EmployeeID
    )
    SELECT
```

```
20      Employees.Id,
21      Employees.LastName,
22      revenue_by_employee.total_revenue,
23      sales_by_employee.sales_count,
24      revenue_by_employee.total_revenue/sales_by_employee.sales_count AS
25
    avg_revenue_per_sale
26      FROM revenue_by_employee
27      JOIN sales_by_employee ON revenue_by_employee.EmployeeId =
    sales_by_employee.EmployeeId
28      JOIN Employees ON revenue_by_employee.EmployeeId = Employees.Id
29      ORDER BY total_revenue DESC
30
```