SQL-ANALYTICS-II

SQL CTE (Common Table Expression)

SQL Temporary Tables

Assoc. Prof. Mahammad Sharifov

1. SQL CTE (Common Table Expression)

A Common Table Expression (CTE) is the result set of a query which **exists temporarily** and for use only within the context of a larger query. Much like a derived table, the result of a **CTE** is not stored and exists only for the duration of the query.

CTEs**, like database views and derived tables, enable users to more easily write and maintain complex queries via increased readability and simplification. **This reduction in complexity is achieved by deconstructing ordinarily complex queries into simple blocks to be used, and reused, if necessary, in rewriting the query.

Example use cases include:

- Needing to reference a derived table multiple times in a single query
- An alternative to creating a view in the database
- Performing the same calculation multiple times over across multiple query components

1. SQL CTE (Common Table Expression)

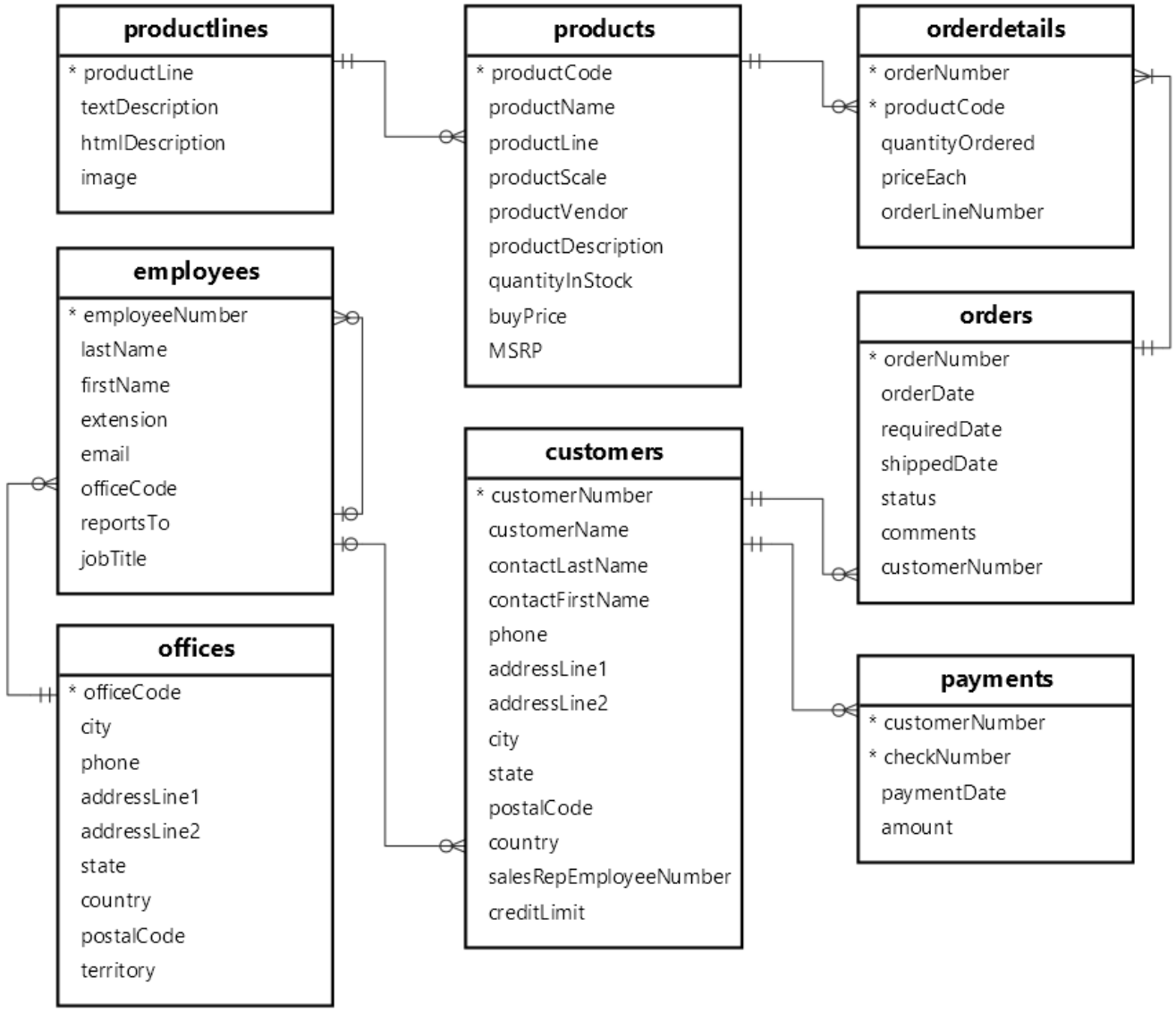
The structure of a CTE includes the name, an optional column list, and a query that defines the CTE. After you define a CTE, you can use like a view in the *SELECT, INSERT, UPDATE, DELETE,* or *CREATE VIEW* statement.

The following illustrates the basic syntax of a CTE:

```
WITH cte_name (column_list) AS (
        query
)
SELECT * FROM cte_name;
```

1. SQL CTE (Common Table Expression)

- **The structure of database:**



1.1. SQL CTE / Example 1

Example-1 : Basic MySQL CTE example
We'll use the customers table from the sample database for demonstration:

```
WITH customers_in_usa AS (
    SELECT
        customerName, state
    FROM
        customers
    WHERE
        country = 'USA'
) SELECT
    customerName
 FROM
    customers_in_usa
 WHERE
    state = 'CA'
 ORDER BY customerName;
```

How it works:

- First, define a CTE with the name customers_in_usa that stores the customer name and state of customers in the USA. The defining query retrieves data from the customer's table.
- Second, select the customers located in California from the CTE.

1.1. SQL CTE / Example 2

Example-2 : Getting top sales using a CTE
We'll use the orders, orderdetails, employees table from the sample database for demonstration:

How it works:

- The following example uses a CTE to retrieve the **top 5 sales representatives** based on their total sales in the year 2003.
- Second, join the CTE with the employees table to include the first and last names of the sales representatives.

```
+----------------+-----------+-----------+-----------+
| employeeNumber | firstName | lastName  | sales     |
+----------------+-----------+-----------+-----------+
|           1165 | Leslie    | Jennings  | 413219.85 |
|           1370 | Gerard    | Hernandez | 295246.44 |
|           1401 | Pamela    | Castillo  | 289982.88 |
|           1621 | Mami      | Nishi     | 267249.40 |
|           1501 | Larry     | Bott      | 261536.95 |
+----------------+-----------+-----------+-----------+
5 rows in set (0.02 sec)
```

1.1. SQL CTE / Example 2

Example-2 : Getting top sales using a CTE

```
WITH topsales2003 AS (
    SELECT
        salesRepEmployeeNumber employeeNumber,
        SUM(quantityOrdered * priceEach) sales
    FROM
        orders
            INNER JOIN
        orderdetails USING (orderNumber)
            INNER JOIN
        customers USING (customerNumber)
    WHERE
        YEAR(shippedDate) = 2003
        AND status = 'Shipped'
    GROUP BY salesRepEmployeeNumber
    ORDER BY sales DESC
    LIMIT 5
)
```

```
SELECT
    employeeNumber,
    firstName,
    lastName,
    sales
FROM
    employees
        JOIN
    topsales2003 USING (employeeNumber);
```

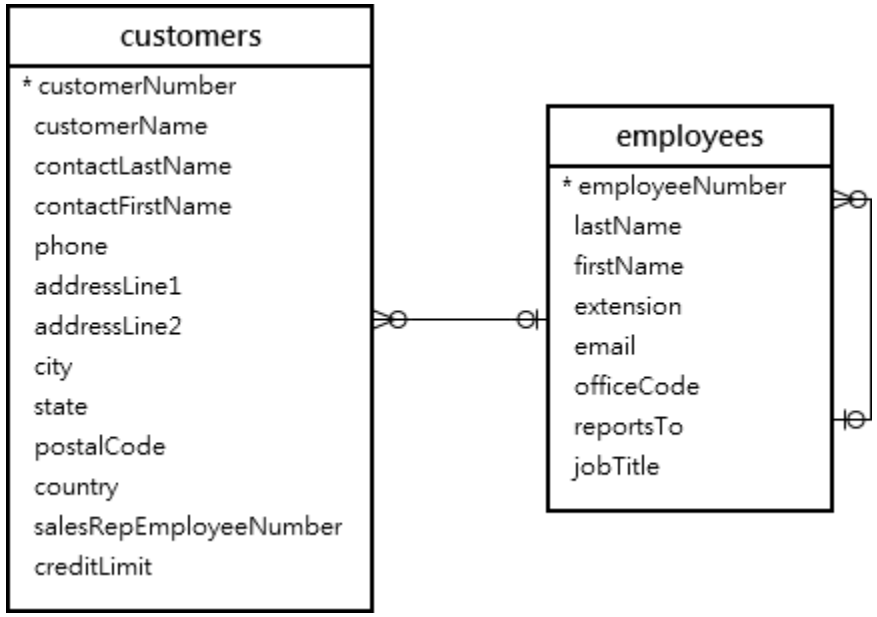
1.2. Multiple CTEs / Example 3

Example-3 : The following example uses multiple CTEs to map the customers with their respective sales representatives.

We'll use the customers and employees from the sample database

How it works:

- **CTE salesrep**: Select employeeNumber and concatenate the firstName and lastName columns to create a column named salesrepName, and include only employees with the job title 'Sales Rep'.
- **CTE customer_salesrep**: selects customerName and salesrepName by joining the customers table with the salesrep CTE based on the common column employeeNumber.
- **Main query**: Select all columns from the customer_salesrep CTE.



1.2. Multiple CTEs / Example 3

Example-3 : The following example uses multiple CTEs to map the customers with their respective sales representatives

```
WITH salesrep AS (
    SELECT
        employeeNumber,
        CONCAT(firstName, ' ', lastName) AS salesrepName
    FROM
        employees
    WHERE
        jobTitle = 'Sales Rep’
) ,
customer_salesrep AS (
    SELECT
        customerName, salesrepName
    FROM
        customers
            INNER JOIN
        salesrep ON employeeNumber =
salesrepEmployeeNumber
)
```

```
SELECT
    *
FROM
    customer_salesrep
ORDER BY customerName;
```

```
+-----------------------------------+-----------------+
| customerName                      | salesrepName    |
+-----------------------------------+-----------------+
| Alpha Cognac                      | Gerard Hernandez |
| American Souvenirs Inc            | Foon Yue Tseng  |
| Amica Models & Co.                | Pamela Castillo |
| Anna's Decorations, Ltd           | Andy Fixter     |
| Atelier graphique                 | Gerard Hernandez |
| Australian Collectables, Ltd      | Andy Fixter     |
| Australian Collectors, Co.        | Andy Fixter     |
| Australian Gift Network, Co       | Andy Fixter     |
...
```

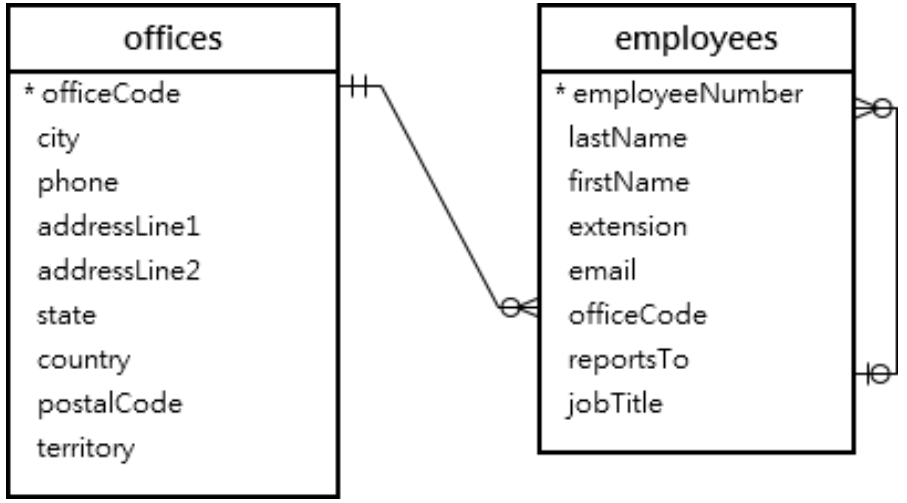
1.3. Multiple CTEs and JOIN / Example 4

Example-4 : The following example is creating two CTEs and joining them to get the Sales Representatives located in the USA, including their office information.

We'll use the offices and employees from the sample database

How it works:

- **CTE e:** Retrieve employees whose job title is Sales Rep.
- **CTE o:** Retrieve offices located in the USA.
- **Main query:** Joins the CTE e and o using the officeCode column.



1.3. Multiple CTEs and JOIN / Example 4

Example-4 : The following example

```
WITH E AS (
  SELECT
    *
  FROM
    employees
  WHERE
    jobTitle = 'Sales Rep'
),
O AS (
  SELECT
    *
  FROM
    offices
  WHERE
    country = 'USA'
)
```

```
SELECT
  firstName, lastName, city, state,
  postalCode
FROM
  E
  INNER JOIN O USING (officeCode);
```

```
+-----------+-----------+---------------+-------+------------+
| firstName | lastName  | city          | state | postalCode |
+-----------+-----------+---------------+-------+------------+
| Leslie    | Jennings  | San Francisco | CA    | 94080      |
| Leslie    | Thompson  | San Francisco | CA    | 94080      |
| Julie     | Firrelli  | Boston        | MA    | 02107      |
| Steve     | Patterson | Boston        | MA    | 02107      |
| Foon Yue  | Tseng     | NYC           | NY    | 10022      |
| George    | Vanauf    | NYC           | NY    | 10022      |
+-----------+-----------+---------------+-------+------------+
6 rows in set (0.00 sec)
```

CTE Exercises / Exercise 1

Using CTE try to solve the following sub-tasks:

- Get sum of orders by each customer
- Get number of orders by each customer
- List of all products with productName (inline, comma separated)

orderNumber	allProducts	sumOrders	countOrders
10100	1917 Grand Touring Sedan,1911 Ford Town Car,1932 Alfa	10223.83	4
10101	1932 Model A Ford J-Coupe,1928 Mercedes-Benz SSK,1939 Chevrolet Deluxe Coupe	10549.01	4
10102	1937 Lincoln Berline,1936 Mercedes-Benz 500K Special	5494.78	2
10103	1952 Alpine Renault 1300, 1962 LanciaA Delta 16V, 1958	50218.95	16
10104	1969 Corvair Monza,1957 Chevy Pickup,1998 Chrysler	40206.20	13
10105	1972 Alfa Romeo GTA,2001 Ferrari Enzo,1969 Ford Falcon	53959.21	15
10106	1980s Black Hawk Helicopter,P-51-D Mustang,1999	52151.81	18

1. CTE / Exercises 1-5

1. By each productCode(name) to find SUM of orders
2. By each productLine to find COUNT of products
3. By each orderNumber to find MIN price of product, MAX price of product! (use window functions)
4. By each orederNumber to display RANK and DENSE_RANK (use window functions)
5. By each productLine to display RANK of buyPrice, MIN and MAX of buyPrices as well.

2. SQL Temporary Tables / Definition

In SQL, a temporary table is a special type of table that allows you to store a temporary result set, which you can reuse several times in a single session.

A MySQL temporary table has the following features:

- A temporary table is created by using CREATE TEMPORARY TABLE statement.
- MySQL removes the temporary table automatically when the session ends, or the connection is terminated. Also, you can use the DROP TABLE statement to remove a temporary table
- A temporary table is only available and accessible to the client that creates it. Different clients can create temporary tables with the same name without causing errors because only the client that creates the temporary table can see it. However, in the same session, two temporary tables cannot share the same name.

2. SQL Temporary Tables / Definition

- A temporary table can have the same name as a regular table in a database. For example, if you create a temporary table named employees in the sample database, the existing employees table becomes inaccessible. Every query you issue against the employees table is now referring to the temporary table employees.

Note : Even though a temporary table can have the same name as a regular table, it is not recommended. Because this may lead to confusion and potentially cause an unexpected data loss.

For example, if the connection to the database server is lost and you reconnect to the server automatically, you cannot differentiate between the temporary table and the regular one.

Then, you may issue a **DROP TABLE** statement to remove the permanent table instead of the temporary table, which is not expected.

To avoid this issue, you can use the **DROP TEMPORARY TABLE**

2. SQL Temporary Tables / Syntax

CASE 1 : The syntax of the CREATE TEMPORARY TABLE statement is like the syntax of the CREATE TABLE statement except for the TEMPORARY keyword:

```
CREATE TEMPORARY TABLE table_name(
   column1 datatype constraints,
   column1 datatype constraints,
   ...,
   table_constraints
) ;
```

CASE 2 : To create a temporary table whose **structure is based on an existing table**, we use the following syntax:

```
CREATE TEMPORARY TABLE
   temp_table_name
SELECT * FROM original_table
LIMIT 0 ;
```

2. SQL Temporary Tables / Example 1

STEP 1 : First, create a new temporary table called credits that stores customers' credits:

```
CREATE TEMPORARY TABLE credits(
    customerNumber INT PRIMARY KEY,
    creditLimit DEC(10, 2)
) ;
```

STEP 2 : Then, insert rows from the customers table into the temporary table credits:

```
INSERT INTO credits(customerNumber,
    creditLimit)
SELECT
    customerNumber,
    creditLimit
FROM
    customers
WHERE
    creditLimit > 0 ;
```

2. SQL Temporary Tables / Example 1

TESTING TEMPORARY TABLE : CREDITS

STEP 1 : First, write simple query with select statement to prove existence of temporary ‘credits’ table

STEP 2 : Second, switch to different database, then come back to current database (classicmodels) and again test existence of ‘credits’ table

STEP 3 : At the end, quit current session and reconnect to database again to test existence of ‘credits’ temporary table

2. SQL Temporary Tables / Example 2

CASE 2 : **Creating a temporary table whose structure is based on a query example:**
The following example creates a temporary table that stores the top 10 customers by revenue.
The structure of the temporary table is derived from a SELECT statement:

```
CREATE TEMPORARY TABLE  top_customers
SELECT p.customerNumber,
            c.customerName,
            ROUND ( SUM(p.amount), 2) sales
FROM payments p
INNER JOIN customers c ON
c.customerNumber = p.customerNumber
GROUP BY p.customerNumber
ORDER BY sales DESC
LIMIT 10 ;
```

```
SELECT
    customerNumber,
    customerName,
    sales
FROM
    top_customers
ORDER BY sales ;
```

2. SQL Temporary Tables / Dropping

DROPPING TEMPORARY TABLE : You can use the DROP TABLE statement to remove temporary tables however it is good practice to add the TEMPORARY keyword as follows:

```
DROP TEMPORARY TABLE  table_name;
```

2. SQL Temporary Tables / Exercises

EXERCISE 1 :

Use the *customers* and *employees* from the sample database.

- The following exercise should use **multiple CTEs** to map the customers with their respective sales representatives;
- Create **Temporary Table** from combining 2 CTE's by select statement

EXERCISE 2 :

- Create **Temporary Table** by selecting (partition by) each *orderNumber* displaying with *MIN price* of order, *MAX price* of order;
- Use select statement to display results **by grouping** *orderNumber* with *MIN price of order* and *MAX price* of orders;
- Use select statement to select results from both **Temporary Table** and **Permanent Table** : *oredrNumber, productName, MIN price* and *MAX price*