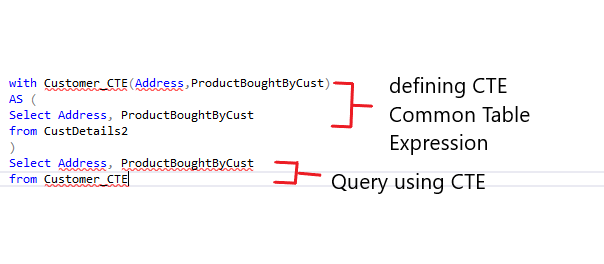
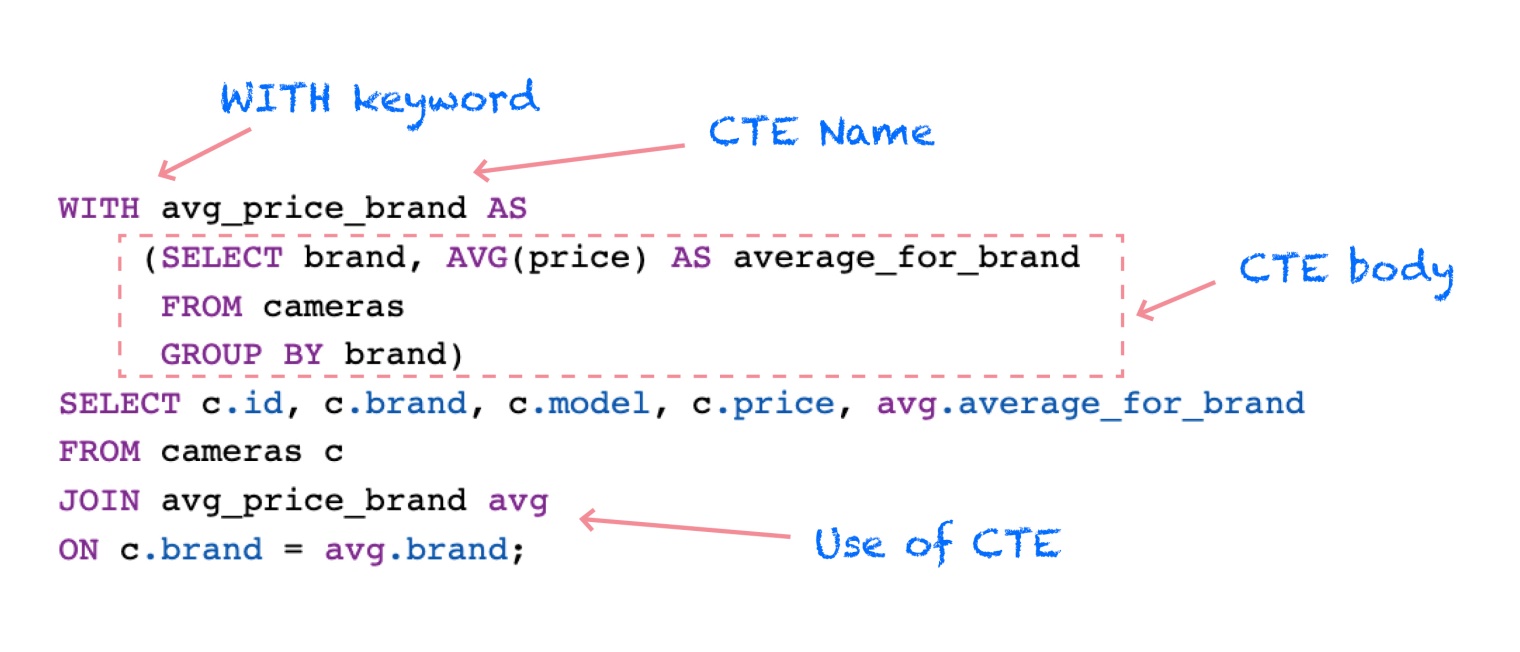
**SQL\_Common Table Expressions (CTE)**

The **Common Table Expressions (CTE)** was introduced into standard SQL in order to simplify various classes of SQL Queries for which a derived table was just unsuitable. The common table expression (CTE) is a temporary named result set that you can reference within a SELECT, INSERT, UPDATE, or DELETE statement.





**Syntax:**

We can define CTEs by adding a WITH clause directly before SELECT, INSERT, UPDATE, DELETE, or MERGE statement. The WITH clause can include one or more CTEs separated by commas. The following syntax can be followed:

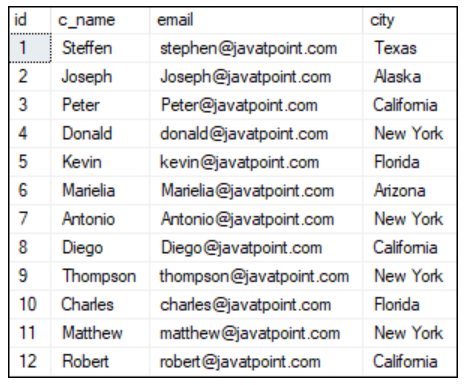
**WITH** cte\_name (column\_names)

**AS** (query)

**SELECT** \* **FROM** cte\_name;

**Example:**

Customer table



**Q. In this example, create CTE named customers\_in\_newyork, returning the three columns customer name, email, and state. The CTE customers\_in\_newyork should return all customers who live in New York State.**

**WITH** customers\_in\_NewYork

**AS** (**SELECT** \* **FROM** customer **WHERE** state = 'New York')

**SELECT** c\_name, email, state **FROM** customers\_in\_NewYork;

Here, we can see that the result returns only that customer information that is located in New York State.



**Q1. Write a query to create a CTE that stores name and age of employees and retrieve it from the CTE.**

with nameage(name,age) as

(select name,age from employee)

select name,age from nameage;

**Q2. Write a query to create a CTE that stores name and salary of employees displaying ‘greater than equal to 35000’ and ‘less than equal to 35000’ as per the salary for marketing department employees and retrieve it from the CTE.**

with salarylimit(name,salary) as

(select name,if(salary>=35000,'greater than equal to 35000','less than 35000') as 'salary\_range' from employee

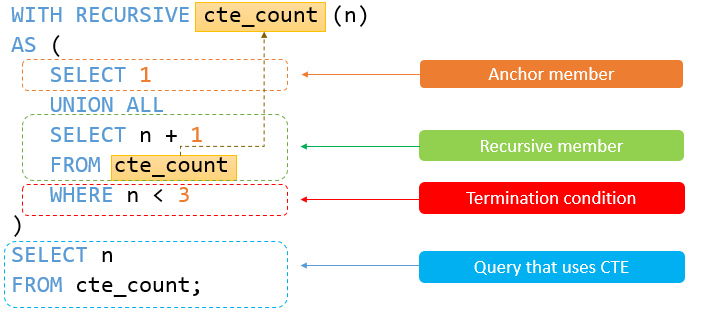
where department='Marketing')

select name,salary from salarylimit;

**Recursive CTE:**

A **recursive CTE** is a sub query which refers to itself using its own name.

* The recursive CTEs are defined using **WITH RECURSIVE** clause.
* There should be a terminating condition to recursive CTE.
* The recursive CTEs are used for series generation and traversal of hierarchical or tree-structured data.



**Syntax:**

WITH RECURSIVE

cte\_name [(col1, col2, ...)]

AS ( subquery )

Select col1, col2, .. from cte\_name;

**Example:**

**Q. Create a recursive CTE that returns weekdays from Monday to Saturday.**

WITH recursive cte\_numbers(n, weekday)

AS (

SELECT

0,

DATE\_FORMAT(CURDATE() + INTERVAL 0 DAY, '%W')

UNION ALL

SELECT

n + 1,

DATE\_FORMAT(CURDATE() + INTERVAL n + 1 DAY, '%W')

FROM

cte\_numbers

WHERE n < 6

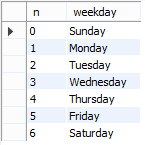
)

SELECT

n, weekday

FROM

cte\_numbers;



**Q3. Write a query to print a series of first 5 odd numbers using recursive CTE. (serial no., odd no.)**

with recursive

odd\_no(sr\_no,odd\_n) as

(select 1,1

union all

select sr\_no+1,odd\_n+2 from odd\_no where sr\_no<5)

select \* from odd\_no;

