POSTGRESQL - WITH CLAUSE

In PostgreSQL, the WITH query provides a way to write auxiliary statements for use in a larger query. It helps in breaking down complicated and large queries into simpler forms, which are easily readable. These statements, which are often referred to as Common Table Expressions or CTEs, can be thought of as defining temporary tables that exist just for one query.

The WITH query being CTE query, is particularly useful when subquery is executed multiple times. It is equally helpful in place of temporary tables. It computes the aggregation once and allows us to reference it by its name (may be multiple times) in the queries.

The WITH clause must be defined before it is used in the query.

Syntax:

The basic syntax of WITH query is as follows:

```
WITH
   name_for_summary_data AS (
     SELECT Statement)
   SELECT columns
   FROM name_for_summary_data
   WHERE conditions <=> (
                       SELECT column
                       FROM name_for_summary_data)
   [ORDER BY columns]
```

Where *name_for_summary_data* is the name given to the WITH clause. The name_for_summary_data can be the same as an existing table name and will take precedence.

You can use data-modifying statements (INSERT, UPDATE or DELETE) in WITH. This allows you to perform several different operations in the same query.

Recursive WITH

Recursive WITH or Hierarchical queries, is a form of CTE where a CTE can reference to itself, i.e., a WITH query can refer to its own output, hence the name recursive.

Example

Consider the table COMPANY having records as follows:

```
testdb# select * from COMPANY;
 id | name  | age | address    | salary
----+-------+-----+------------+--------
  1 | Paul  |  32 | California|  20000
  2 | Allen |  25 | Texas      |  15000
  3 | Teddy |  23 | Norway     |  20000
  4 | Mark  |  25 | Rich-Mond  |  65000
  5 | David |  27 | Texas      |  85000
  6 | Kim   |  22 | South-Hall|  45000
  7 | James |  24 | Houston    |  10000
(7 rows)
```

Now, let us write a query using the WITH clause to select the records from the above table, as follows:

```
With CTE AS
(Select
 ID
, NAME
, AGE
```

```
, ADDRESS
, SALARY
FROM COMPANY )
Select * From CTE;
```

Above PostgreSQL statement will produce the following result:

```
id | name  | age | address   | salary
----+-------+-----+-----------+--------
  1 | Paul  |  32 | California|  20000
  2 | Allen |  25 | Texas     |  15000
  3 | Teddy |  23 | Norway    |  20000
  4 | Mark  |  25 | Rich-Mond |  65000
  5 | David |  27 | Texas     |  85000
  6 | Kim   |  22 | South-Hall|  45000
  7 | James |  24 | Houston   |  10000
(7 rows)
```

Now, let us write a query using the RECURSIVE keyword along with the WITH clause, to find the sum of the salaries less than 20000, as follows:

```
WITH RECURSIVE t(n) AS (
    VALUES (0)
    UNION ALL
    SELECT SALARY FROM COMPANY WHERE SALARY < 20000
)
SELECT sum(n) FROM t;
```

Above PostgreSQL statement will produce the following result:

```
  sum
-------
 25000
(1 row)
```

Let us write a query using data modifying statements along with the WITH clause, as follows. First create a table COMPANY1 similar to the table COMPANY. The query in the example, effectively moves rows from COMPANY to COMPANY1. The DELETE in WITH deletes the specified rows from COMPANY, returning their contents by means of its RETURNING clause; and then the primary query reads that output and inserts it into COMPANY1 TABLE:

```
CREATE TABLE COMPANY1(
   ID INT PRIMARY KEY     NOT NULL,
   NAME           TEXT    NOT NULL,
   AGE            INT     NOT NULL,
   ADDRESS        CHAR(50),
   SALARY         REAL
);

WITH moved_rows AS (
    DELETE FROM COMPANY
    WHERE
        SALARY >= 30000
    RETURNING *
)
INSERT INTO COMPANY1 (SELECT * FROM moved_rows);
```

Above PostgreSQL statement will produce the following result:

```
INSERT 0 3
```

Now, the records in the tables COMPANY and COMPANY1 are as follows:

```
testdb=# SELECT * FROM COMPANY;
 id | name  | age |  address   | salary
```

```
----+-------+-----+------------+--------
  1 | Paul  |  32 | California |  20000
  2 | Allen |  25 | Texas      |  15000
  3 | Teddy |  23 | Norway     |  20000
  7 | James |  24 | Houston    |  10000
(4 rows)


testdb=# SELECT * FROM COMPANY1;
 id | name  | age | address | salary
----+-------+-----+-------------+--------
  4 | Mark  |  25 | Rich-Mond   |  65000
  5 | David |  27 | Texas       |  85000
  6 | Kim   |  22 | South-Hall  |  45000
(3 rows)
```