**CTE - common table expression**

A common table expression is a temporary result set which you can reference within another SQL statement including SELECT, INSERT, UPDATE or DELETE.

Common Table Expressions are temporary in the sense that they only exist during the execution of the query.

eg : WITH cte\_name (column\_list) AS (

CTE\_query\_definition

)

statement;

**Recursive CTE**

A recursive query is a query that refers to a recursive CTE. The recursive queries are useful in many situations such as querying hierarchical data like organizational structure, bill of materials, etc.

The recursive term is one or more CTE query definitions joined with the non-recursive term using the UNION or UNION ALL operator. The recursive term references the CTE name itself.

eg:

WITH RECURSIVE cte\_name AS(

CTE\_query\_definition -- non-recursive term

UNION [ALL]

CTE\_query definion -- recursive term

) SELECT \* FROM cte\_name;

**Views**

A view is a named query that provides another way to present data in the database tables. A view is defined based on one or more tables which are known as base tables. When you create a view, you basically create a query and assign a name to the query. Therefore, a view is useful for wrapping a commonly used complex query.

eg:

CREATE VIEW usa\_cities AS SELECT

city,

country\_id

FROM

city

WHERE

country\_id = 103;

DELETE

FROM

usa\_cities

WHERE

city = 'San Jose';

**MY REFERENCE**

A view is a database object that is of a stored query. A view can be accessed as a virtual table in PostgreSQL. In other words, a PostgreSQL view is a logical table that represents data of one or more underlying tables through a SELECT statement.

Creating Views

The PostgreSQL views are created using the CREATE VIEW statement. The PostgreSQL views can be created from a single table, multiple tables, or another view.

The basic CREATE VIEW syntax is as follows −

CREATE [TEMP | TEMPORARY] VIEW view\_name AS

SELECT column1, column2.....

FROM table\_name

WHERE [condition];

You can include multiple tables in your SELECT statement in very similar way as you use them in normal PostgreSQL SELECT query. If the optional TEMP or TEMPORARY keyword is present, the view will be created in the temporary space. Temporary views are automatically dropped at the end of the current session.

Example

Consider, the [COMPANY](https://www.tutorialspoint.com/postgresql/company.sql) table is having the following records −

id | name | age | address | salary

----+-------+-----+------------+--------1 | Paul | 32 | California | 200002 | Allen | 25 | Texas | 150003 | Teddy | 23 | Norway | 200004 | Mark | 25 | Rich-Mond | 650005 | David | 27 | Texas | 850006 | Kim | 22 | South-Hall | 450007 | James | 24 | Houston | 10000

Now, following is an example to create a view from COMPANY table. This view would be used to have only few columns from COMPANY table −

testdb=#

CREATE VIEW COMPANY\_VIEW AS

SELECT ID, NAME, AGE

FROM COMPANY;

Now, you can query COMPANY\_VIEW in a similar way as you query an actual table. Following is the example −

testdb=# SELECT \* FROM COMPANY\_VIEW;

This would produce the following result −

id | name | age

----+-------+-----

1 | Paul | 32

2 | Allen | 25

3 | Teddy | 23

4 | Mark | 25

5 | David | 27

6 | Kim | 22

7 | James | 24

(7 rows)

**Materialized views**

A materialized view is a database object that contains the results of a query. The FROM clause of the query can name tables, views, and other materialized views. Collectively these objects are called master tables (a replication term) or detail tables (a data warehousing term).

**Transactions**

A transaction is a unit of work that is performed against a database. Transactions are units or sequences of work accomplished in a logical order, whether in a manual fashion by a user or automatically by some sort of a database program. A transaction is the propagation of one or more changes to the database.

**Schema migrations**

Database migrations, also known as schema migrations, database schema migrations, or simply migrations, are controlled sets of changes developed to modify the structure of the objects within a relational database.

Migrations help transition database schemas from their current state to a new desired state, whether that involves adding tables and columns, removing elements, splitting fields, or changing types and constraints.