Let's Name the Returned CTE Column

```
WITH source AS (
        SELECT 1 AS col1
)
SELECT * FROM source;
 col1
------
    1
```

The CTE returned column is *source.col1*.

The Column Can Also Be Named in the WITH Clause

```
WITH source (col1) AS (
        SELECT 1
)
SELECT * FROM source;
 col1
------
    1
```

Columns Can Be Renamed

```
WITH source (col2) AS (
        SELECT 1 AS col1
)
SELECT col2 AS col3 FROM source;
 col3
------
    1
```

The CTE column starts as *col1,* is renamed in the WITH clause as *col2,* and the outer SELECT renames it to *col3*.

Multiple CTE Columns Can Be Returned

```
WITH source AS (
        SELECT 1, 2
)
SELECT * FROM source;
 ?column? | ?column?
----------+----------
        1 |        2
```

UNION Refresher

```
SELECT 1
UNION
SELECT 1;
 ?column?
----------
        1

SELECT 1
UNION ALL
SELECT 1;
 ?column?
----------
        1
        1
```

Possible To Create Multiple CTE Results

```
WITH source AS (
        SELECT 1, 2
),
     source2 AS (
        SELECT 3, 4
)
SELECT * FROM source
UNION ALL
SELECT * FROM source2;
 ?column? | ?column?
----------+----------
        1 |        2
        3 |        4
```

CTE with Real Tables

```
WITH source AS (
        SELECT lanname, rolname
        FROM pg_language JOIN pg_roles ON lanowner = pg_roles.oid
)
SELECT * FROM source;
 lanname  | rolname
----------+----------
 internal | postgres
 c        | postgres
 sql      | postgres
 plpgsql  | postgres
```

CTE Can Be Processed More than Once

```
WITH source AS (
        SELECT lanname, rolname
        FROM pg_language JOIN pg_roles ON lanowner = pg_roles.oid
        ORDER BY lanname
)
SELECT * FROM source
UNION ALL
SELECT MIN(lanname), NULL
FROM source;
 lanname  | rolname
----------+----------
 c        | postgres
 internal | postgres
 plpgsql  | postgres
 sql      | postgres
 c        |
```

CTE Can Be Joined

```
WITH class AS (
        SELECT oid, relname
        FROM pg_class
        WHERE relkind = 'r'
)
SELECT class.relname, attname
FROM pg_attribute, class
WHERE class.oid = attrelid
ORDER BY 1, 2
LIMIT 5;
   relname    |   attname
--------------+--------------
 pg_aggregate | aggfinalfn
 pg_aggregate | aggfnoid
 pg_aggregate | agginitval
 pg_aggregate | aggsortop
 pg_aggregate | aggtransfn
```

Imperative Control With CASE

```
CASE
WHEN condition THEN result
ELSE result
END
```

For example:

```
SELECT col,
    CASE
        WHEN col > 0 THEN 'positive'
        WHEN col = 0 THEN 'zero'
        ELSE 'negative'
    END
FROM tab;
```