# LIMIT and OFFSET

`LIMIT` and `OFFSET` allow you to retrieve just a portion of the rows that are generated by the rest of the query:

```
SELECT select_list
    FROM table_expression
    [LIMIT { number | ALL }] [OFFSET number]
```

If a limit count is given, no more than that many rows will be returned (but possibly less, if the query itself yields less rows). `LIMIT ALL` is the same as omitting the `LIMIT` clause.

`OFFSET` says to skip that many rows before beginning to return rows. `OFFSET 0` is the same as omitting the `OFFSET` clause. If both `OFFSET` and `LIMIT` appear, then `OFFSET` rows are skipped before starting to count the `LIMIT` rows that are returned.

When using `LIMIT`, it is important to use an `ORDER BY` clause that constrains the result rows into a unique order. Otherwise you will get an unpredictable subset of the query's rows. You may be asking for the tenth through twentieth rows, but tenth through twentieth in what ordering? The ordering is unknown, unless you specified `ORDER BY`.

The query optimizer takes `LIMIT` into account when generating a query plan, so you are very likely to get different plans (yielding different row orders) depending on what you give for `LIMIT` and `OFFSET`. Thus, using different `LIMIT`/`OFFSET` values to select different subsets of a query result **will give inconsistent results** unless you enforce a predictable result ordering with `ORDER BY`. This is not a bug; it is an

inherent consequence of the fact that SQL does not promise to deliver the results of a query in any particular order unless `ORDER BY` is used to constrain the order.

The rows skipped by an `OFFSET` clause still have to be computed inside the server; therefore a large `OFFSET` can be inefficient.

Obtenido de: https://www.postgresql.org/docs/8.1/static/queries-limit.html.
Consultado: 27 de enero de 2017