SQL Cheat Sheet

Starter guide for standard SQL syntax used in PostgreSQL

Jason Lee Apr 18, 2020 · 5 min read ★

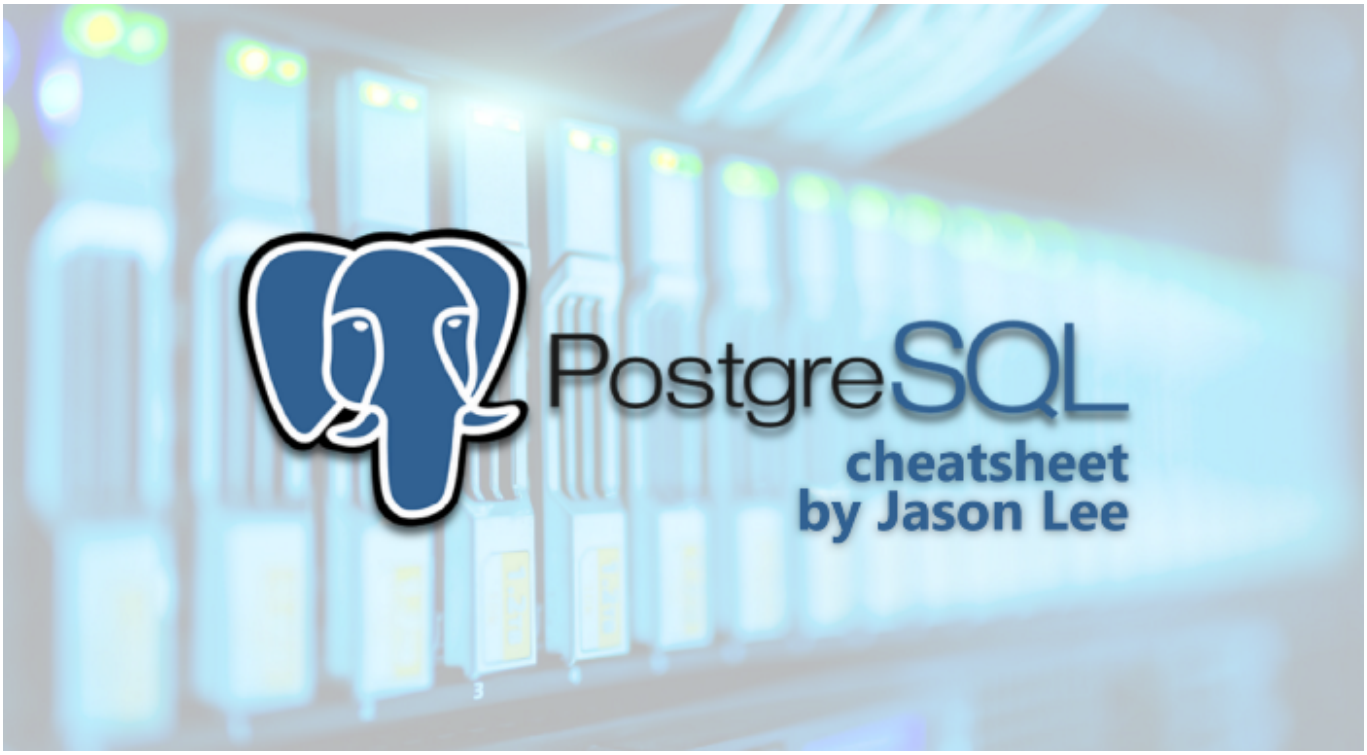

Photo by panumas nikhomkhai from Pexels

SQL is the most important coding language to learn for data analysis. Some might argue Python and R are equally important, but when it comes to the most common tool an Analyst must have it is SQL.

According to Dataquest.io almost all of the biggest names in tech use SQL. Uber, Netflix, Airbnb — the list goes on. Even within companies like Facebook, Google, and Amazon, which have built their own high-performance database systems, data teams use SQL to query data and perform analysis.

Like every language, you need to keep practicing to understand and grasp keep concepts. In my opinion, SQL is one of the easier languages to use once you understand the basic structure of the code. In this article, I share the necessary steps to getting started with SQL querying.

Standard SQL Structure

This is **Part 1** to a series of PostgreSQL cheat sheets and will cover `SELECT`, `FROM`, `WHERE`, `GROUP BY`, `HAVING`, `ORDER BY` and `LIMIT`.

The basic structure of a query pulling results from a single table is as follows.

```
SELECT
        COLUMN_NAME(S)
FROM
        TABLE_NAME
WHERE
        CONDITION
GROUP BY
        COLUMN_NAME(S)
HAVING
        AGGREGATE_CONDITION
ORDER BY
        COLUMN_NAME
LIMIT
        N
```

What is SQL?

SQL (pronounced “ess-que-el”) stands for Structured Query Language. SQL is used to communicate with a database. It is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database or retrieve data from a database.

What is Relational Database Management System (RDBMS)?

An RDBMS organizes data into tables with rows and columns. The term relational means that values within each table have a relationship with each other.

- Rows — also known as records
- Columns — also known as fields, have a descriptive name and specific data type.

What is PostgreSQL?

PostgreSQL is a general-purpose and relational database management system, the most advanced open-source database system.

Other common database management systems are MySQL, Oracle, IBM Db2, and MS Access.

Let’s begin!

SELECT

The SELECT statement is used to select data from a database. The data returned is stored in a result table, called the result-set.

Specific columns

```
SELECT
        COLUMN_1,
        COLUMN_2
FROM
        TABLE_NAME
```

All columns

Using the `*` you can query every column in your table

```
SELECT *
FROM
        TABLE_NAME
```

DISTINCT Columns

Finding all the unique records in a column

```
SELECT
        DISTINCT(COLUMN_NAME)
FROM
        TABLE_NAME
```

COUNT all rows

If you want to know all the values in the entire table use `COUNT(*)` you will get a single number.

```
SELECT
        COUNT(*)
FROM
        TABLE_NAME
```

COUNT DISTINCT values

If you want the number of distinct values in a column using `COUNT` with `DISTINCT` and you will get a number representing the total unique values of a column

```
SELECT
        COUNT (DISTINCT COLUMN_NAME)
FROM
        TABLE_NAME
```

WHERE

Using the `WHERE` the clause, you can create conditions to filter out values you want or don't want.

NOTE — `WHERE` is always used before a `GROUP BY` (More on this later)

```
SELECT *
FROM
        TABLE_NAME
WHERE
        CONDITION
```

Conditions

There are a variety of conditions that can be used in SQL. Below are some examples of a table that consists of students' grades in school. You only need to specify `WHERE` once, for the sake

of the example, I have included `WHERE` in each step.

```
WHERE FIRSTNAME      = 'BOB'      -- exact match
WHERE FIRSTNAME     != 'BOB'     -- everything excluding BOB
WHERE NOT FIRSTNAME  ='BOB'    -- everything excluding BOB

WHERE FIRSTNAME IN ('BOB', 'JASON')       -- either condition is met
WHERE FIRSTNAME NOT IN ('BOB', 'JASON')   -- excludes both values

WHERE FIRSTNAME = 'BOB' AND LASTNAME = 'SMITH'  -- both conditions
WHERE FIRSTNAME = 'BOB' OR FIRSTNAME = 'JASON'  -- either condition

WHERE GRADES > 90           -- greater than 90
WHERE GRADES < 90           -- less than 90
WHERE GRADES  >= 90         -- greater than or equal to 90
WHERE GRADES  <= 90         -- less than or equal to 90

WHERE SUBJECT IS NULL       -- returns values with missing values
WHERE SUBJECT NOT NULL      -- returns values with no missing values
```

Conditions — Wildcards

`LIKE` operator is used in a `WHERE` clause to search for a specified pattern in a column. When you pass the `LIKE` operator in the `''` upper and lower case matters.

There are two wildcards often used in conjunction with the `LIKE` operator:

- `%` - The percent sign represents zero, one, or multiple characters
- `_` - The underscore represents a single character

```
WHERE FIRSTNAME LIKE ‘B%’ -- finds values starting uppercase B

WHERE FIRSTNAME LIKE ‘%b’ -- finds values ending lowercase b

WHERE FIRSTNAME LIKE ‘%an%’ -- find values that have “an” in any
position

WHERE FIRSTNAME LIKE ‘_n%’ -- find values that have “n” in the
second position

WHERE FIRSTNAME LIKE ‘B__%’ -- find values that start with “B” and
have at least 3 characters in length
```

```
WHERE FIRSTNAME LIKE ‘B%b’ -- find values that start with “B” and
end with “b”

WHERE FIRSTNAME LIKE ‘[BFL]’ -- find all values that start with ‘B’,
‘F’ OR ‘L’

WHERE FIRSTNAME LIKE ‘[B-D]’ -- find all values that start with ‘B’,
‘C’, OR ‘D’

WHERE FIRSTNAME LIKE ‘[!BFL]%’ -- find everything exlcusing values
that start with ‘B’, ‘F’ OR ‘L’

WHERE FIRSTNAME NOT LIKE ‘[BFL]%’ -- same as above. excludes values
starting with ‘B’, ‘F’, OR ‘L’

WHERE GRADES BETWEEN 80 and 90 -- find grades between 80 and 90
```

GROUP BY

The `GROUP BY` function helps calculate summary values by the chosen column. It is often used with aggregate functions (`COUNT` , `SUM` , `AVG` , `MAX` , `MIN`).

```
SELECT
        SUBJECT,
        AVG(GRADES)
FROM
        STUDENTS
GROUP BY
        SUBJECT
```

The query above will group each subject and calculate the **average** grades.

```
SELECT
        SUBJECT,
        COUNT(*)
FROM
        STUDENTS
GROUP BY
        SUBJECT
```

The above query will calculate the **number** (count) of students in each subject.

HAVING

The `HAVING` clause is similar to `WHERE` but is catered for **filtering aggregate functions**. The `HAVING` function comes after the `GROUP BY`, in comparison the `WHERE` comes before the `GROUP BY`.

If we wanted to find which subject had an average grade of 90 or more, we could use the following.

```
SELECT
        SUBJECT,
        AVG(GRADES)
FROM
        STUDENTS
GROUP BY
        SUBJECT
HAVING
        AVG(GRADES) >= 90
```

ORDER BY

Using the `ORDER BY` function, you can specify how you want your values sorted. Continuing with the Student tables from earlier.

```
SELECT
        *
FROM
        STUDENTS
ORDER BY
        GRADES DESC
```

When using the `ORDER BY` by default, the sort will be in **ascending** order. If you want to descend, you need to specify `DESC` after the column name.

LIMIT

In Postgres, we can use the `LIMIT` function to control how many rows are outputted in the query. For example, if we wanted to find the top 3 students with the highest grades.

```
SELECT
        *
FROM
        STUDENTS
ORDER BY
        GRADES DESC
LIMIT
        3
```

Since we use `ORDER BY DESC` we have the order of students with the highest grades on top - now limiting it to 3 values, we see the **top 3**.

Overview

Hopefully, you can use this starter guide for standard SQL syntax used when querying data from a single table. There is a lot more you can do in SQL, and I will be sharing more **SQL cheat sheets** that will expand advanced syntax.

If you want to learn specific techniques, check out my other tutorials.