< Previous | Next >

SQL Guide: Getting started

Just as humans use different languages to communicate with others, so do computers. **Structured Query Language** (or **SQL**, often pronounced "sequel") enables data analysts to talk to their databases. SQL is one of the most useful data analyst tools, especially when working with large datasets in tables. It can help you investigate huge databases, track down text (referred to as strings) and numbers, and filter for the exact kind of data you need—much faster than a spreadsheet can.

If you haven't used SQL before, this reading will help you learn the basics so you can appreciate how useful SQL is and how useful SQL queries are in particular. You will be writing SQL queries in no time at all.

What is a query?

A **query** is a request for data or information from a database. When you query databases, you use SQL to communicate your question or request. You and the database can always exchange information as long as you speak the same language.

Every programming language, including SQL, follows a unique set of guidelines known as **syntax**. **Syntax** is the predetermined structure of a language that includes all required words, symbols, and punctuation, as well as their proper placement. As soon as you enter your search criteria using the correct syntax, the query starts working to pull the data you've requested from the target database.

The syntax of every SQL query is the same:

- Use **SELECT** to choose the columns you want to return.
- Use **FROM** to choose the tables where the columns you want are located.
- Use **WHERE** to filter for certain information.

A SQL query is like filling in a template. You will find that if you are writing a SQL query from scratch, it is helpful to start a query by writing the SELECT, FROM, and WHERE keywords in the following format:

```
SELECT

FROM

WHERE
```

Next, enter the table name after the FROM; the table columns you want after the SELECT; and, finally, the conditions you want to place on your query after the WHERE. Make sure to add a new line and indent when adding these, as shown below:

```
SELECT
            Columns you want to look at
FROM
            Table the data lives in
WHERE
            Certain condition is met
```

Following this method each time makes it easier to write SQL queries. It can also help you make fewer syntax errors.

Example of a query

Here is how a simple query would appear in BigQuery, a data warehouse on the Google Cloud Platform.

```
SELECT
        first_name
FROM
        customer_data.customer_name
WHERE
        first_name = 'Tony'
```

The above query uses three commands to locate customers with the first name Tony:

1. **SELECT** the column named **first_name**

2. **FROM** a table named **customer_name** (in a dataset named **customer_data**)
   (The dataset name is always followed by a dot, and then the table name.)

3. But only return the data **WHERE** the first_name is **Tony**

The results from the query might be similar to the following:

first_name
Tony
Tony
Tony

As you can conclude, this query had the correct syntax, but wasn't very useful after the data was returned.

Multiple columns in a query

In real life, you will need to work with more data beyond customers named Tony. Multiple columns that are chosen by the same SELECT command can be indented and grouped together.

If you are requesting multiple data fields from a table, you need to include these columns in your SELECT command. Each column is separated by a comma as shown below:

```
SELECT
        ColumnA,
        ColumnB,
        ColumnC
FROM
        Table where the data lives
WHERE
        Certain condition is met
```

Here is an example of how it would appear in BigQuery:

```
SELECT
        customer_id,
        first_name,
        last_name
FROM
        customer_data.customer_name
WHERE
        first_name = 'Tony'
```

The above query uses three commands to locate customers with the first name Tony.

1. **SELECT** the columns named **customer_id**, **first_name**, and **last_name**

2. **FROM** a table named **customer_name** (in a dataset named **customer_data**)
   (The dataset name is always followed by a dot, and then the table name.)

3. But only return the data **WHERE** the first_name is **Tony**

The only difference between this query and the previous one is that more data columns are selected. The previous query selected first_name only while this query selects customer_id and last_name in addition to first_name. In general, it is a more efficient use of resources to select only the columns that you need. For example, it makes sense to select more columns if you will actually use the additional fields in your WHERE clause. If you have multiple conditions in your WHERE clause, they may be written like this:

```
SELECT
        ColumnA,
        ColumnB,
        ColumnC
FROM
        Table where the data lives
WHERE
        Condition 1
        AND Condition 2
        AND Condition 3
```

Notice that unlike the SELECT command that uses a comma to separate fields/variables/parameters, the WHERE command uses the AND statement to connect conditions. As you become a more advanced writer of queries, you will make use of other connectors/operators such as OR and NOT.

Here is a BigQuery example with multiple fields used in a WHERE clause:

```
SELECT
        customer_id,
```

```
        first_name,
        last_name
FROM
        customer_data.customer_name
WHERE
        customer_id > 0
        AND first_name = 'Tony'
        AND last_name = 'Magnolia'
```

The above query uses three commands to locate customers with a valid (greater than 0) customer ID whose first name is Tony and last name is Magnolia.

1. **SELECT** the columns named **customer_id**, **first_name**, and **last_name**

2. **FROM** a table named **customer_name** (in a dataset named **customer_data**)
   (The dataset name is always followed by a dot, and then the table name.)

3. But only return the data **WHERE** customer_id is greater than **0**, first_name is **Tony**, and last_name is **Magnolia**.

Note that one of the conditions is a logical condition that checks to see if customer_id is greater than zero.

If only one customer is named Tony Magnolia, the results from the query could be:

customer_id	first_name	last_name
1967	Tony	Magnolia

If more than one customer has the same name, the results from the query could be:

customer_id	first_name	last_name
1967	Tony	Magnolia
7689	Tony	Magnolia

Key takeaway

The most important thing to remember is how to use SELECT, FROM, and WHERE in a query. Queries with multiple fields will become simpler after you practice writing your own SQL queries later in the program.

Mark as completed

Like   Dislike   Report an issue