Secret identities: The importance of aliases

In this reading, you will learn about using aliasing to simplify your SQL queries. **Aliases** are used in SQL queries to create temporary names for a column or table. Aliases make referencing tables and columns in your SQL queries much simpler when you have table or column names that are too long or complex to make use of in queries. Imagine a table name like special_projects_customer_negotiation_mileages. That would be difficult to retype every time you use that table. With an alias, you can create a meaningful nickname that you can use for your analysis. In this case “special_projects_customer_negotiation_mileages” can be aliased to simply “mileage.” Instead of having to write out the long table name, you can use a meaningful nickname that you decide.

Basic syntax for aliasing

Aliasing is the process of using aliases. In SQL queries, aliases are implemented by making use of the AS command. The basic syntax for the AS command can be seen in the following query for aliasing a table:

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

Notice that AS is preceded by the table name and followed by the new nickname. It is a similar approach to aliasing a column:

```
SELECT column_name AS alias_name
FROM table_name;
```

In both cases, you now have a new name that you can use to refer to the column or table that was aliased.

Alternate syntax for aliases

If using AS results in an error when running a query because the SQL database you are working with doesn't support it, you can leave it out. In the previous examples, the alternate syntax for aliasing a table or column would be:

- FROM table_name alias_name
- SELECT column_name alias_name

The key takeaway is that queries can run with or without using AS for aliasing, but using AS has the benefit of making queries more readable. It helps to make aliases stand out more clearly.

Aliasing in action

Let’s check out an example of a SQL query that uses aliasing. Let’s say that you are working with two tables: one of them has employee data and the other one has department data. The FROM statement to alias those tables could be:

```
FROM work_day.employees AS employees
```

These aliases still let you know exactly what is in these tables, but now you don’t have to manually input those long table names. Aliases can be really helpful for long, complicated queries. It is easier to read and write your queries when you have aliases that tell you what is included within your tables.

For more information

If you are interested in learning more about aliasing, here are some resources to help you get started:

- **SQL Aliases** **:** This tutorial on aliasing is a really useful resource to have when you start practicing writing queries and aliasing tables on your own. It also demonstrates how aliasing works with real tables.
- **SQL Alias** **:** This detailed introduction to aliasing includes multiple examples. This is another great resource to reference if you need more examples.
- **Using Column Aliasing** **:** This is a guide that focuses on column aliasing specifically. Generally, you will be aliasing entire tables, but if you find yourself needing to alias just a column, this is a great resource to have bookmarked.

Mark as completed

Like **Dislike** **Report an issue**