# SQL Style Guide and Naming Conventions

- Intro
- Why we need code style?
    - Query without formatting
    - Formatted with Sqlfluf
- DBeaver configuration
- General Guidance
- Postgres and Greenplum specific
- Best Practices
- Commenting
- Naming Conventions
    - Tables
    - Views
    - Columns
- SQLFluff

## Intro

SQL Style Guide and Naming Conventions are very important. You can identify the importance by reviewing those two examples. We can see the same query, but one of them was not formatted and another was automatically formatted by Sqlfluf.

Different projects and different databases have different Style Guide and Naming Conventions, and this particular document was created for Postgres and Greenplum. But it can be used for all other databases.

There are a lot of different rules (>100) but do not panic! Programmers love to make the computer do their work, and they create a lot of tools, and we will use one amazing tool, The SQL Linter for Humans SQLFluff.

This article was written based on GitLab SQL Style Guide

## Why we need code style?

Please take a look at two queries. They both produce the same result. But what query is better from your point of view?

### Query without formatting

```
WITH final AS( SELECT
-- This is a single line comment
my_data.field_1 AS detailed_field_1, my_data.field_2 AS detailed_field_2, my_data.detailed_field_3,
DATE_TRUNC('month', some_cte.date_field_at) AS date_field_month, some_cte.data_by_row['id']::NUMBER AS
id_field,
IFF(my_data.detailed_field_3 > my_data.field_2, TRUE, FALSE) AS is_boolean, CASE WHEN my_data.
cancellation_date IS NULL
AND my_data.expiration_date IS NOT NULL THEN my_data.expiration_date WHEN my_data.cancellation_date IS NULL
THEN my_data.start_date + 7
-- There is a reason for this number
ELSE my_data.cancellation_date END AS adjusted_cancellation_date,
COUNT(*) AS number_of_records, SUM(some_cte.field_4) AS field_4_sum, MAX(some_cte.field_5)
AS field_5_max FROM my_data LEFT JOIN some_cte ON my_data.id = some_cte.id WHERE my_data.field_1 = 'abc'
AND (my_data.field_2 = 'def' OR my_data.field_2 = 'ghi') GROUP BY 1, 2, 3, 4, 5, 6 HAVING COUNT(*) > 1 ORDER
BY 8 DESC)
SELECT * FROM final
```

## Formatted with Sqlfluf

this script still has broken code rules, but it can be easily read by any developer without broken eyes

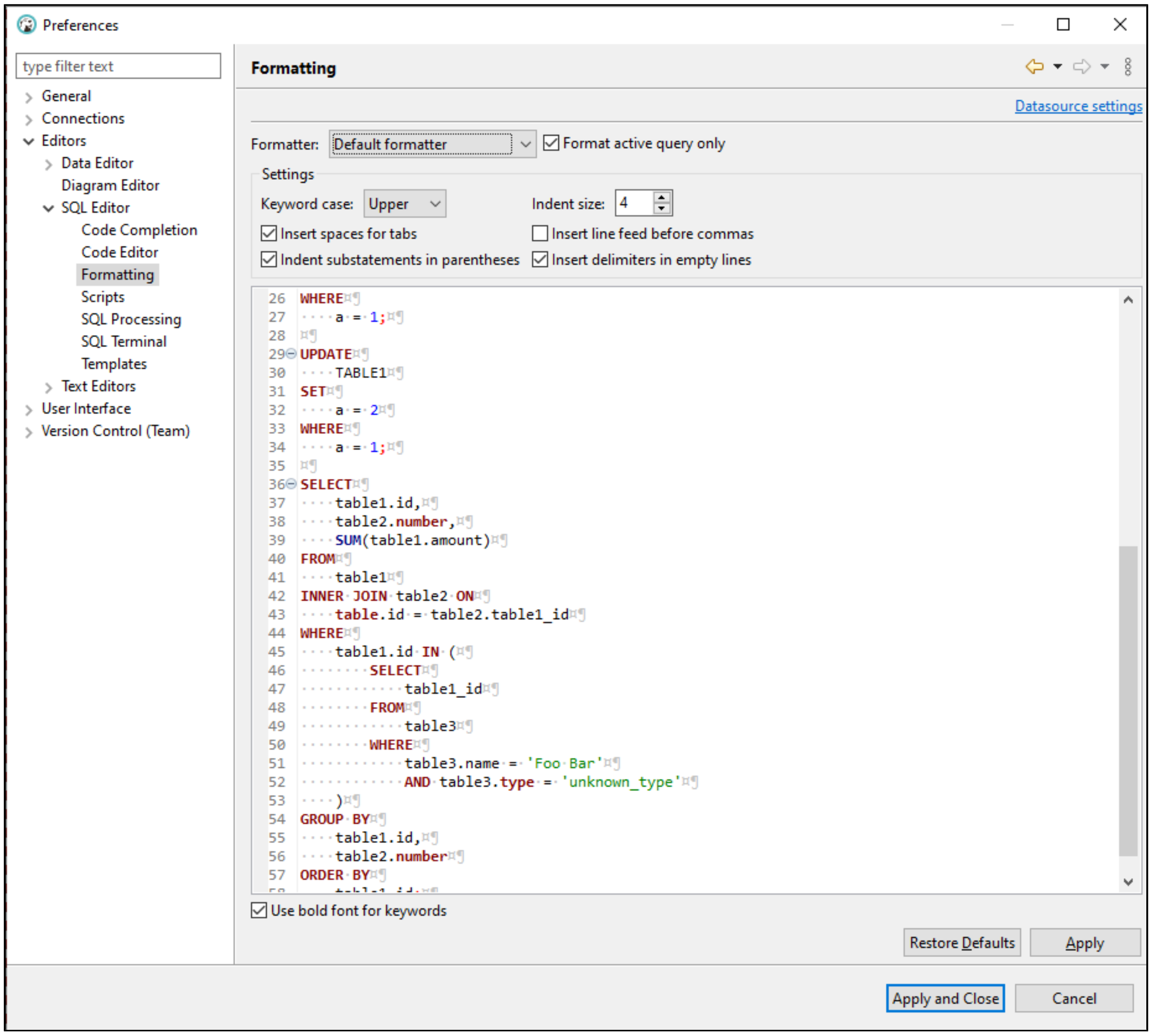
```
WITH
    final AS (

        SELECT
            -- This is a single line comment
            my_data.field_1 AS detailed_field_1,
            my_data.field_2 AS detailed_field_2,
            my_data.detailed_field_3,
            DATE_TRUNC('month', some_cte.date_field_at) AS date_field_month,
            some_cte.data_by_row['id']::NUMBER AS id_field,
            IFF(my_data.detailed_field_3 > my_data.field_2, TRUE, FALSE) AS is_boolean,
            CASE
                WHEN my_data.cancellation_date IS NULL
                    AND my_data.expiration_date IS NOT NULL
                    THEN my_data.expiration_date
                WHEN my_data.cancellation_date IS NULL
                    THEN my_data.start_date + 7 -- There is a reason for this number
                ELSE my_data.cancellation_date
            END AS adjusted_cancellation_date,
            COUNT(*) AS number_of_records,
            SUM(some_cte.field_4) AS field_4_sum,
            MAX(some_cte.field_5) AS field_5_max
        FROM my_data
            LEFT JOIN some_cte
                ON my_data.id = some_cte.id
        WHERE my_data.field_1 = 'abc'
            AND (my_data.field_2 = 'def' OR my_data.field_2 = 'ghi')
        GROUP BY 1, 2, 3, 4, 5, 6
        HAVING COUNT(*) > 1
        ORDER BY 8 DESC
    )

SELECT *
FROM final
```

## DBeaver configuration

Before we start reviewing different rules, please setup the following settings in DBeaver



Preferences

type filter text

- General
- Connections
- Editors
  - Data Editor
  - Diagram Editor
  - SQL Editor
    - Code Completion
    - Code Editor
    - Formatting
    - Scripts
    - SQL Processing
    - SQL Terminal
    - Templates
  - Text Editors
- User Interface
- Version Control (Team)

**Formatting**

Datasource settings

Formatter: Default formatter ☑ Format active query only

Settings

Keyword case: Upper    Indent size: 4

☑ Insert spaces for tabs    ☐ Insert line feed before commas

☑ Indent substatements in parentheses    ☑ Insert delimiters in empty lines

```
WHERE
    a = 1;

UPDATE
    TABLE1
SET
    a = 2
WHERE
    a = 1;

SELECT
    table1.id,
    table2.number,
    SUM(table1.amount)
FROM
    table1
INNER JOIN table2 ON
    table.id = table2.table1_id
WHERE
    table1.id IN (
        SELECT
            table1_id
        FROM
            table3
        WHERE
            table3.name = 'Foo Bar'
            AND table3.type = 'unknown_type'
    )
GROUP BY
    table1.id,
    table2.number
ORDER BY
```

☑ Use bold font for keywords

Restore Defaults   Apply

Apply and Close   Cancel

## General Guidance

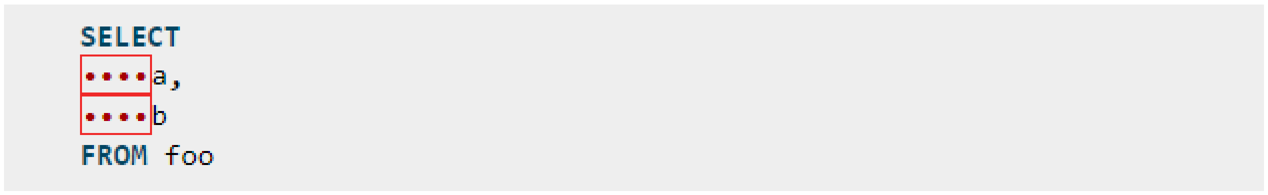
- Do not optimize for fewer lines of code, new lines are cheap, but brain time is expensive.
- Be consistent. Even if you are not sure of the best way to do something do it the same way throughout your code, it will be easier to read and make changes if they are needed.
- Be explicit. Defining something explicitly will ensure that it works the way you expect, and it is easier for the next person, which may be you, when you are explicit in SQL.

## Postgres and Greenplum specific

- Keywords must be in UPPER case (ex SELECT, FROM, JOIN etc,)
- All database objects names must be in lower case and follow **snake_case** (ex process_log, user_id)

## Best Practices

- No tabs should be used - only spaces. Your editor should be setup to convert tabs to spaces - see our onboarding template for more details. Change the indentation to use a multiple of four spaces. This example also assumes that the `indent_unit` config value is set to `space`.



```
SELECT
••••a,
••••b
FROM foo
```

- Wrap long lines of code, between 80 and 100, to a new line.
- Do not use the `USING` command in joins because it produces inaccurate results in Snowflake. Create an account to view the [forum discussion on this topic.]()
- Understand the difference between the following related statements and use appropriately:
    - `UNION ALL` and `UNION`
    - `LIKE` and `ILIKE`
    - `NOT` and `!` and `<>`
    - `DATE_PART()` and `DATE_TRUNC()`
- Use the `AS` operator when aliasing a column or table.
- Prefer `DATEDIFF` to inline additions `date_column + interval_column`. The function is more explicit and will work for a wider variety of date parts.
- Prefer `!=` to `<>`. This is because `!=` is more common in other programming languages and reads like "not equal" which is how we're more likely to speak.
- Prefer `LOWER(column) LIKE '%match%'` to `column ILIKE '%Match%'`. This lowers the chance of stray capital letters leading to an unexpected result.
- Prefer `WHERE` to `HAVING` when either would suffice.

## Commenting

- When making single line comments in a model use the `--` syntax
- When making multi-line comments in a model use the `/* */` syntax
- Respect the character line limit when making comments. Move to a new line or to the model documentation if the comment is too long
- Calculations made in SQL should have a brief description of what's going on and, if available, a link to the handbook defining the metric (and how it's calculated)
- Instead of leaving **`TODO`** comments, create new issues for improvement

## Naming Conventions

### Tables

- Use a collective name. For example **`staff`** or **`employee, material etc`**
- Do not prefix with **`tbl`** or any other such descriptive prefix or Hungarian notation.
- Never give a table the same name as one of its columns and vice versa.

### Views

- prefix view name with  (v_staff,  v_material)
- Use a collective name. For example, v_staff or v_**`employee, etc`**
- Never give a view the same name as one of its columns and vice versa.

### Columns

- An ambiguous field name such as id, `name`, or `type` should always be prefixed by what it is identifying or naming:

```
-- Preferred
SELECT
    id AS account_id,
    name AS account_name,
    type AS account_type,
    ...
-- vs
-- Not Preferred
SELECT
    id,
    name,
    type,
    ...
```

- Always use the singular name.
- Where possible, avoid simply using `id` as the primary identifier for the table.
- Do not add a column with the same name as its table and vice versa.
- Always use lowercase except where it may make sense not to such as proper nouns.

- All field names should be [snake-cased]:
- Boolean field names should start with `has_`, `is_`, or `does_`:
- Timestamps should end with `_at` and should always be in UTC.
- Dates should end with `_date`.
- Avoid keywords like `date` or `month` as a column name.
- System columns would be better to start with sys_ (sys_created_at, sys_created_by_user_id)

Uniform columns suffixes

The following suffixes have a universal meaning, ensuring the columns can be read and understood easily from SQL code. Use the correct suffix where appropriate.

- `_id`—a unique identifier such as a column that is a primary key.
- `_status`—flag value or some other status of any type such as `publication_status`.
- `_total`—the total or sum of a collection of values.
- _num—denotes the field contains any kind of number.
- _name—signifies a name such as `first_name`.
- _seq—contains a contiguous sequence of values.
- `_date`—denotes a column that contains the date of something.
- `_tally`—a count.
- _at - Timestamps
- _type - denotes types
- `_size`—the size of something such as a file size or clothing.
- `_addr`—an address for the record could be physical or intangible such as `ip_addr`.

## SQLFluff

SQLFLuff is a SQL linter that works with many tools. We use it to define the basic structure and style of the SQL that we write and move the review of that structure and style into the hands of the authors.

`Please review SQLfluff page for instruction how to use it`

SQLFluff includes a `fix` command that will apply fixes to rule violations when possible. Not all rule violations are automatically fixable; therefore, you are encouraged to run the `lint` command after using the `fix` command to ensure that all rule violations have been resolved.

- SQLFluff Documentation
- SQLFluff Default configuration

Links

GitLab SQL Style Guide

Brooklyn Data Co. SQL style guide

SQL style guide by Simon Holywell

- SQL Style Guide and Naming Conventions
- Athena: Learned lessons