

Use SQLFluff in VS Code

SQLFluff is a SQL Linter



The screenshot shows the SQLFluff website homepage. At the top, there's a navigation bar with links for Home, Source, Docs, and Blog. The main heading is 'SQLFluff' with a logo featuring a brush. Below it, the tagline 'The SQL Linter for humans.' is displayed. A row of statistics shows 'downloads 987k/month', 'pypi v2.1.2', 'license MIT', and 'stars 6.3k'. A paragraph describes SQLFluff as a dialect-flexible and configurable SQL linter designed for ELT applications, mentioning its compatibility with Jinja templating and dbt, and its auto-fix capabilities. Two code blocks are shown: the first contains a SQL query with linting errors, and the second shows the command-line output of the linter, including error messages like 'Unqualified reference' and 'Qualified reference'.

SQLFluff

[Home](#) [Source](#) [Docs](#) [Blog](#)

SQLFluff

The SQL Linter for humans.

downloads 987k/month pypi v2.1.2 license MIT stars 6.3k

SQLFluff is a dialect-flexible and configurable SQL linter. Designed with ELT applications in mind, SQLFluff also works with Jinja templating and dbt. SQLFluff will auto-fix most linting errors, allowing you to focus your time on what matters.

```
1 SELECT
2     this_column,
3     my_table.THAT_COLUMN AS this_name,
4     some_number *10 AS a_bigger_number
5 from my_table
```

```
$ sqlfluff lint test.sql --dialect ansi
== [test.sql] FAIL
L:  2 | P:  5 | RF03 | Unqualified reference 'this_column' found in single
    | table select. [references.consistent]
L:  3 | P:  5 | RF03 | Qualified reference 'my_table.THAT_COLUMN' found in
```

It checks your code for *lint*

SQLFluff is a dialect-flexible and configurable SQL linter. Designed with ELT applications in mind, **SQLFluff** also works with Jinja templating and dbt. **SQLFluff** will auto-fix most linting errors, allowing you to focus your time on what matters.

```
1 | SELECT
2 |     this_column,
3 |     my_table.THAT_COLUMN AS this_name,
4 |     some_number *10 AS a_bigger_number
5 | from my_table
```

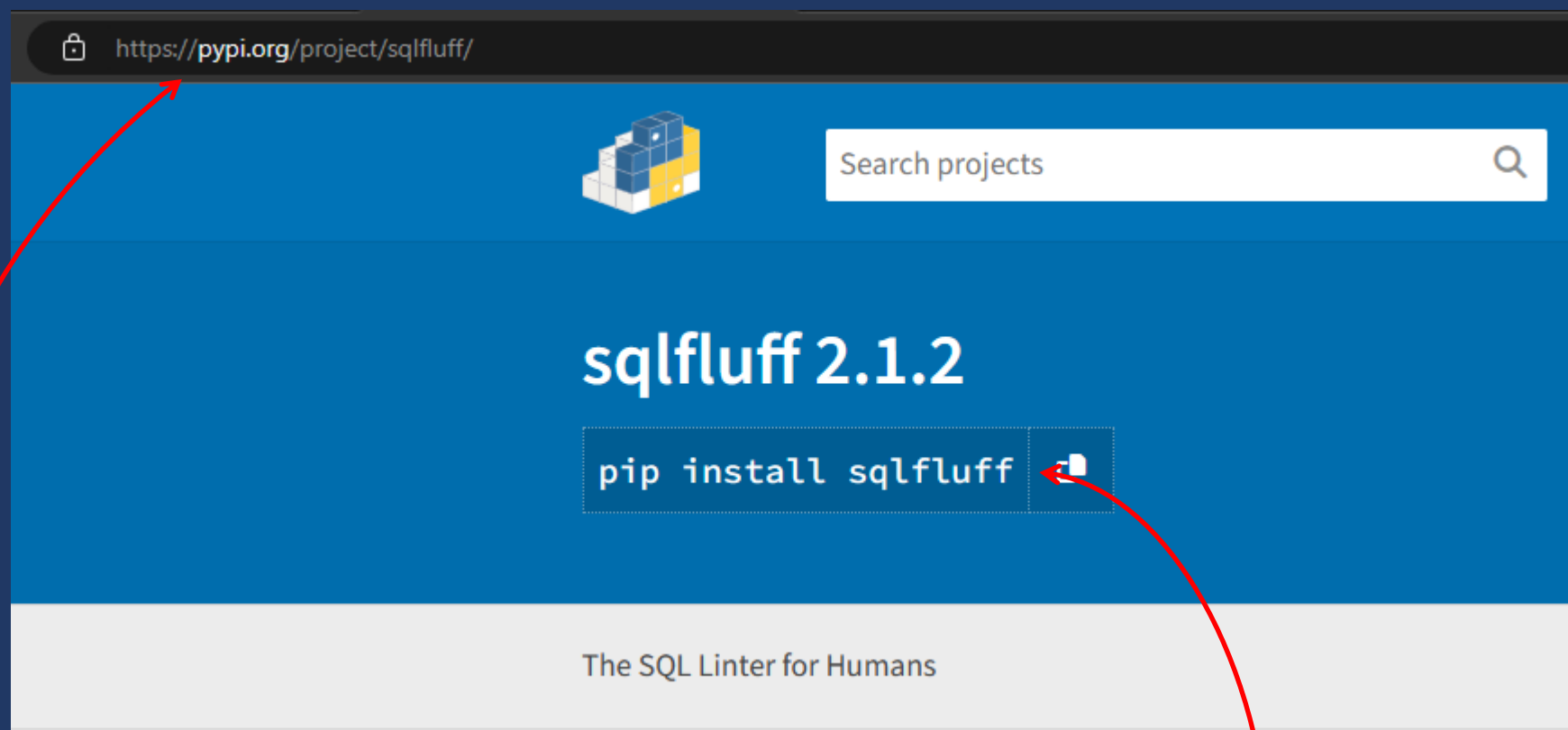
```
$ sqlfluff lint test.sql --dialect ansi
== [test.sql] FAIL
L:   2 | P:   5 | RF03 | Unqualified reference 'this_column' found in single
      |       |       | table select. [references.consistent]
L:   3 | P:   5 | RF03 | Qualified reference 'my_table.THAT_COLUMN' found in
      |       |       | single table select which is inconsistent with previous
      |       |       | references. [references.consistent]
L:   3 | P:  14 | CP02 | Unquoted identifiers must be consistently lower case.
      |       |       | [capitalisation.identifiers]
L:   4 | P:   1 | LT02 | Expected indent of 4 spaces.
      |       |       | [layout.indent]
L:   4 | P:   3 | RF03 | Unqualified reference 'some_number' found in single
      |       |       | table select. [references.consistent]
L:   4 | P:  16 | LT01 | Expected single whitespace between binary operator '*'
      |       |       | and numeric literal. [layout.spacing]
L:   5 | P:   1 | CP01 | Keywords must be consistently upper case.
      |       |       | [capitalisation.keywords]
```

i.e. fluff /
unnecessary
fabric which can
be removed

We can use it to
check the quality
of our SQL

We can define
many rules to
enforce and
check our SQL
automatically
against each one

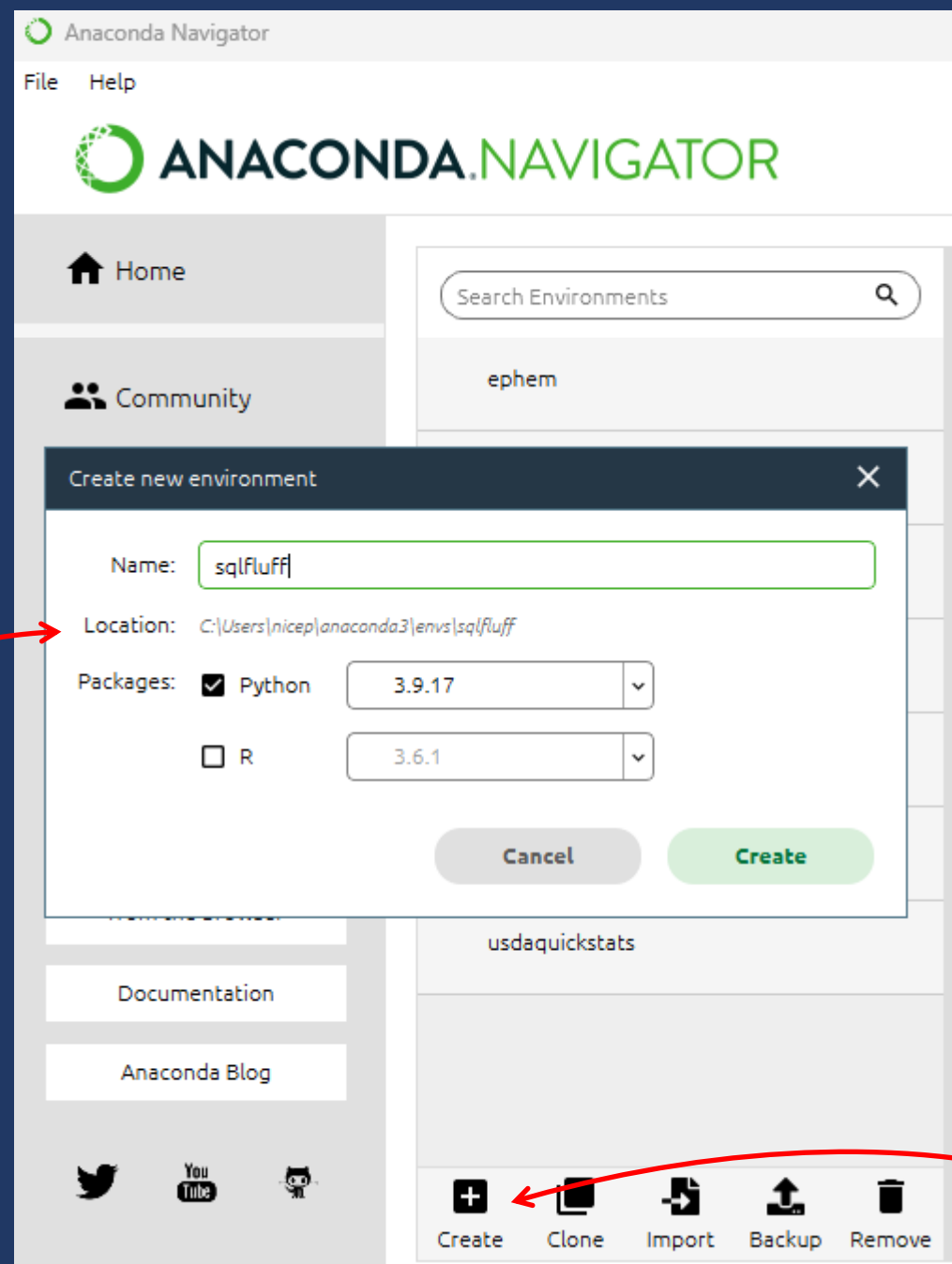
It's a Python package



It's available on PyPI – the official Python Package Index

We can install it into a Python environment using pip

For example, create a new environment in Anaconda



2

Enter a name and select the version of Python to install

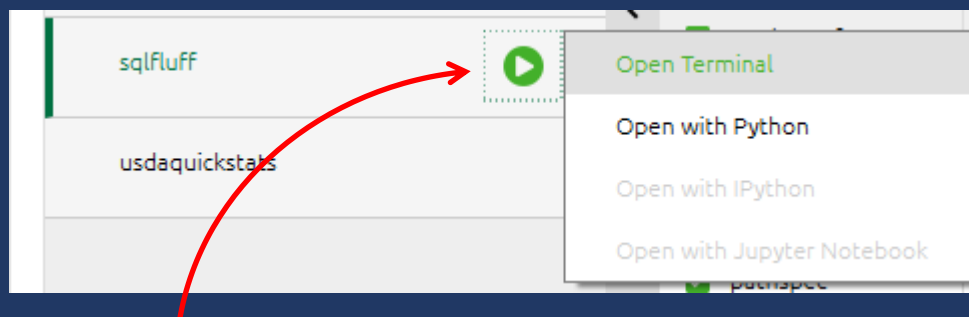


This is just one way of creating an environment. Other ways are available or if you prefer, you can work without an environment (not recommended)

1

From the Environments tab, hit Create

Install sqlfluff from the Anaconda prompt



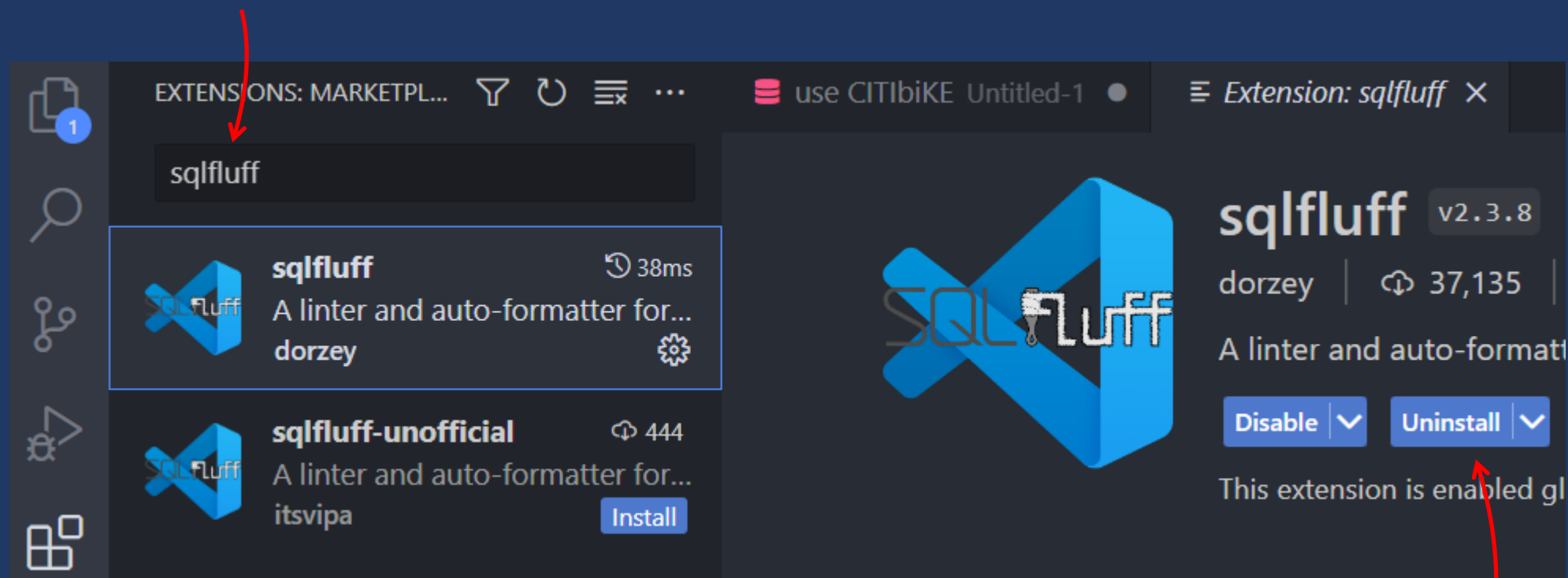
- 1 Hit the play button then 'Open Terminal'

```
(sqlfluff) C:\Users\ >pip install sqlfluff|
```

- 2 From the Anaconda prompt, use **pip install sqlfluff** to install the package

Similarly, install the sqlfluff extension

2 Search for sqlfluff



1 Select the extensions tab


3 Hit this button to install the extension (it will say **Install** when not installed)

Edit *settings.json* in your project folder



Create the file if it doesn't exist

Extension: sqlfluff ✕



sqlfluff v2.3.8
dorzey | 37,135 | ★★★★★ (6)
A linter and auto-formatter for SQLfluff, a popular SQL dialect.

Disable ▾ Uninstall ▾ ⚙️

This extension is enabled globally.

DETAILS FEATURE CONTRIBUTIONS CHANGELOG RUNTIME STATUS

Code command palette and type in `settings.json`. Select **Preferences: Open Settings**. Then, you can add any of the following configuration options to `settings.json`.

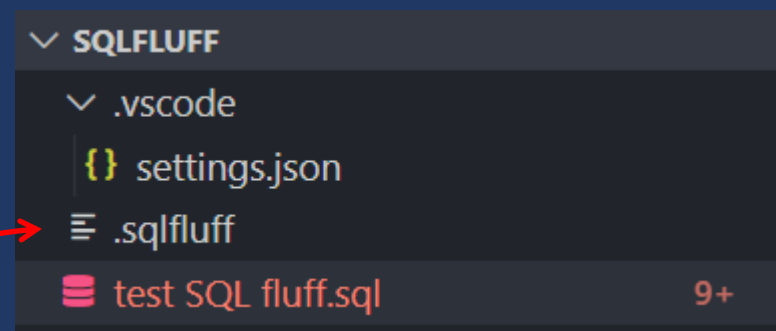
```
"sqlfluff.config": "${workspaceFolder}/.sqlfluff",  
"sqlfluff.dialect": "mysql",  
"sqlfluff.env.environmentVariables": [
```

Scroll down on the extension page, then copy the supplied JSON into your `settings.json` file

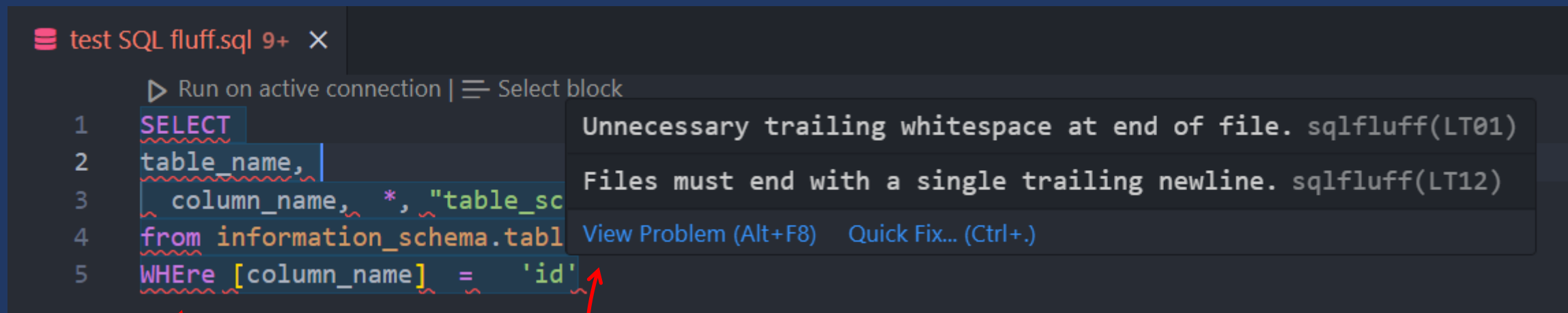
```
test SQL fluff.sql 9+ {} settings.json  
1 {  
2   "sqlfluff.config": "${workspaceFolder}/.sqlfluff",  
3   "sqlfluff.dialect": "tsql",  
4   "sqlfluff.env.environmentVariables": [  
5     {  
6       "key": "example_key",  
7       "value": "example_value"  
8     }  
9   ],  
10  "sqlfluff.env.customDotEnvFiles": [  
11    "${workspaceFolder}/example.env"  
12  ],  
13  "sqlfluff.env.useDotEnvFile": true,  
14  "sqlfluff.excludeRules": [  
15  ],  
16  "sqlfluff.executablePath": "sqlfluff".
```


Create an empty text file called *.sqlfluff* in your project folder

You can leave the file empty for now



Create a new .sql file and type some ugly SQL



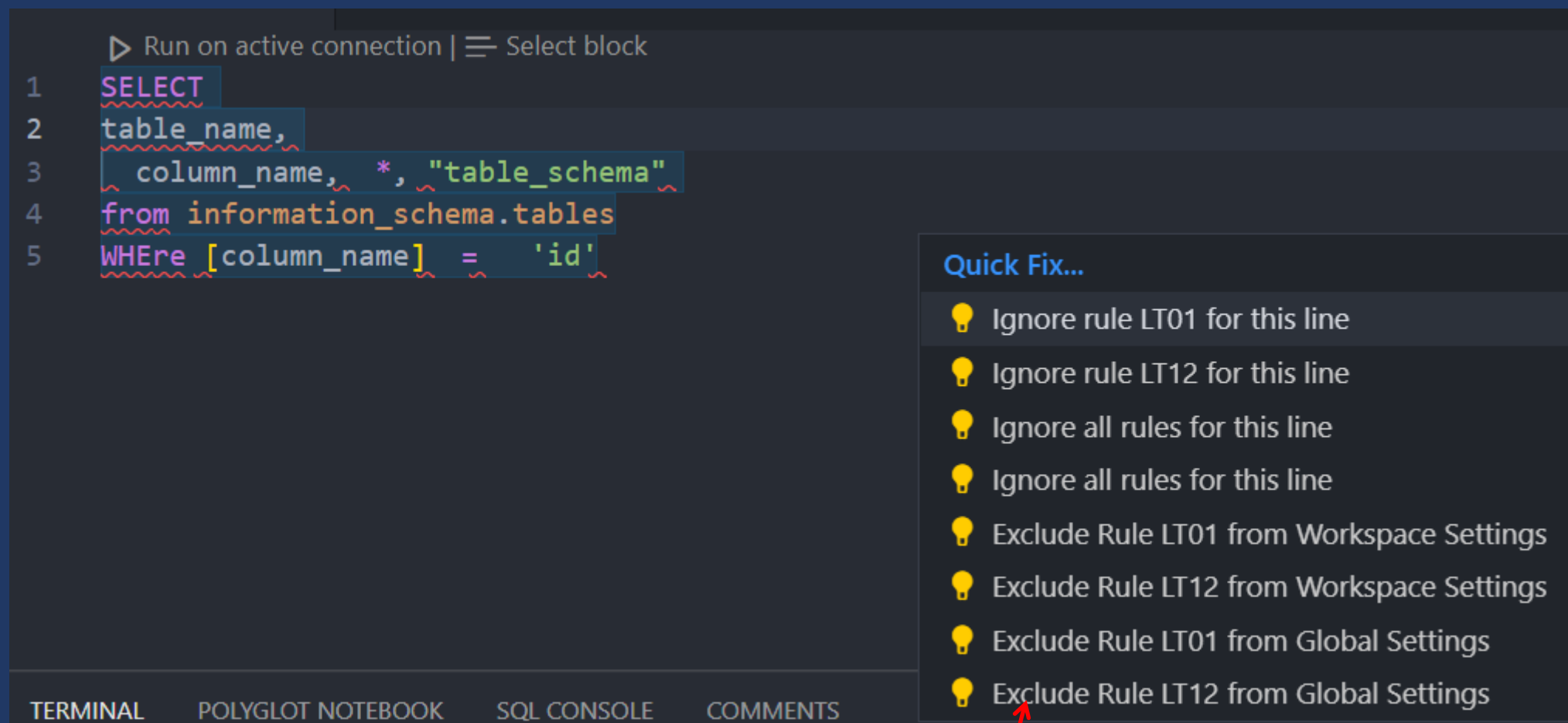
```
test SQL fluff.sql 9+ X
▶ Run on active connection | ≡ Select block
1 SELECT
2 table_name,
3 column_name, *, "table_sc
4 from information_schema.tabl
5 WHEre [column_name] = 'id'
```

Unnecessary trailing whitespace at end of file. sqlfluff(LT01)
Files must end with a single trailing newline. sqlfluff(LT12)
View Problem (Alt+F8) Quick Fix... (Ctrl+.)

SQL which violates the rules will be underlined

Hovering over an underline will show a pop-up describing which rule(s) have been violated

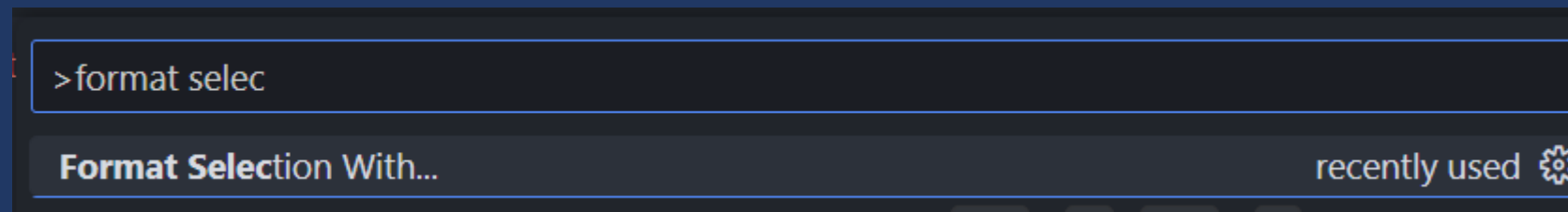
Optionally, hover over a rule and click “Quick Fix”



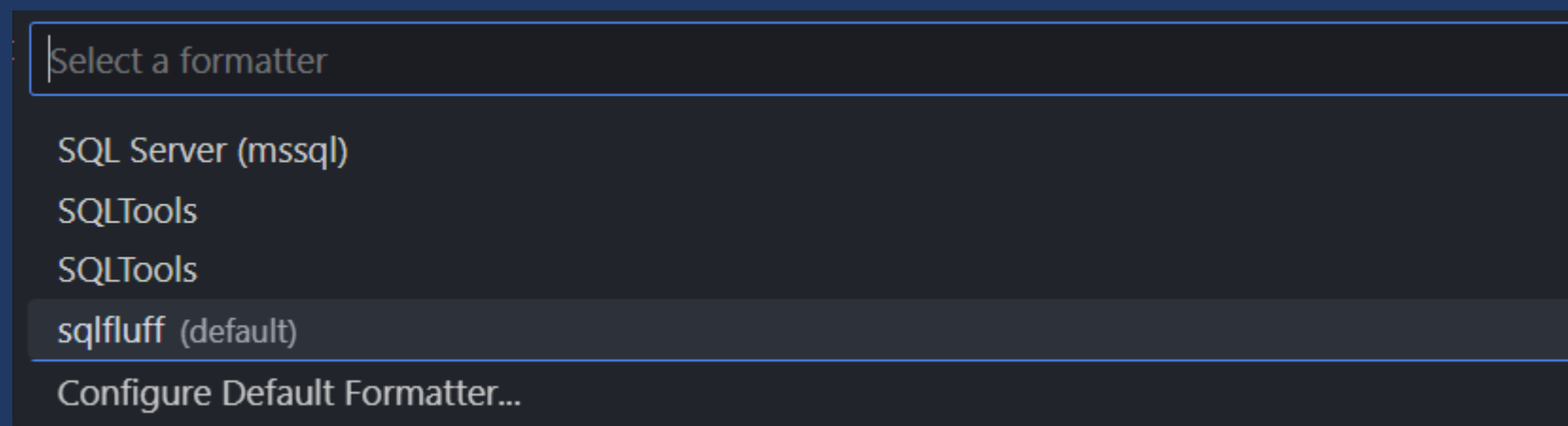
Optionally, select an option to remediate the problem.

Alternatively, we can auto-fix the problems - 1

Ctrl+Shift+P and search for “Format selection with...”

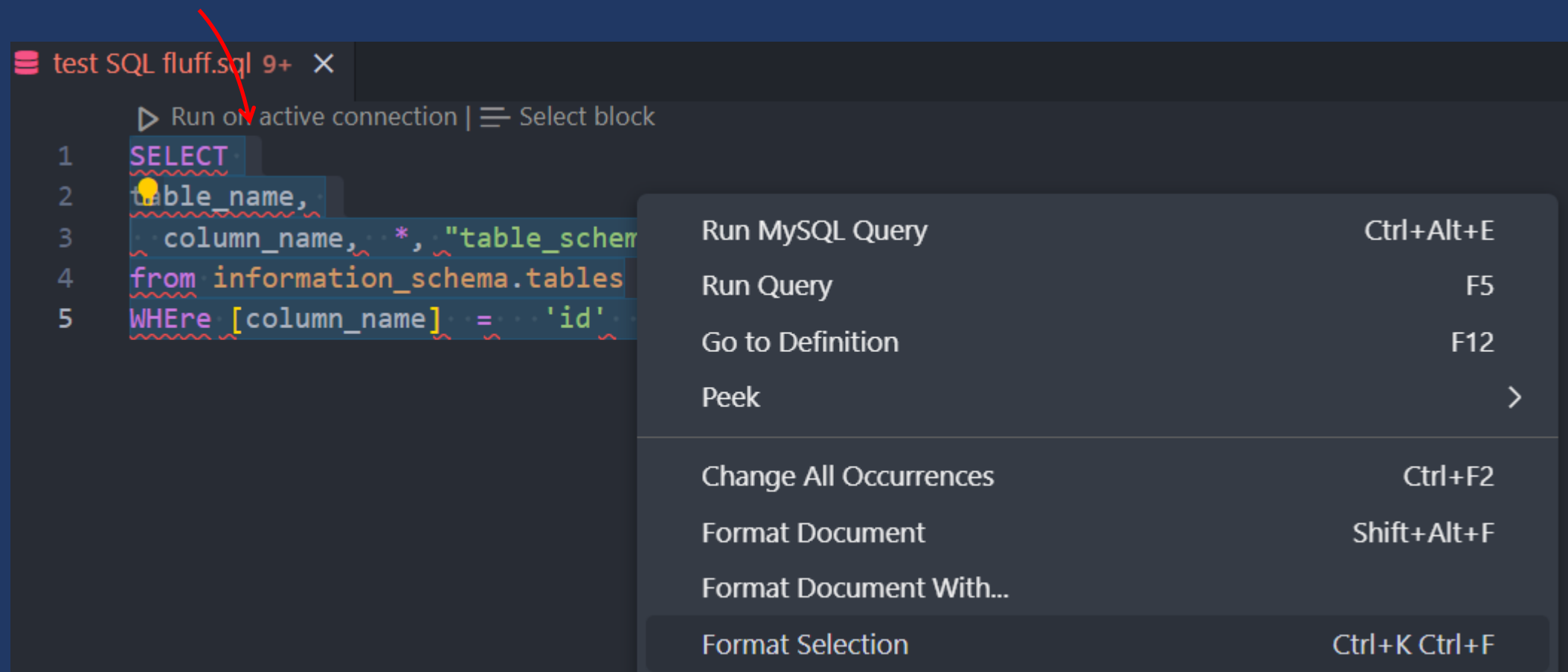


Ensure sqlfluff is configured as the default formatter



Alternatively, we can auto-fix the problems - 2

Select the code you want to fix

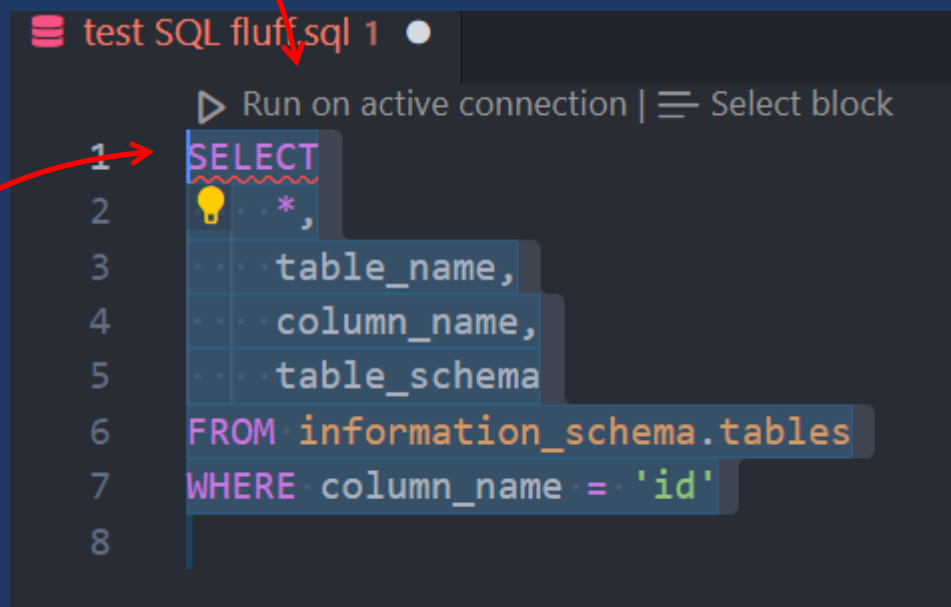


Right-click and use "Format Selection" (alternatively, Ctrl+K Ctrl+F)

Alternatively, we can auto-fix the problems - 3

After formatting, the code is reformatted and *most* rules are fixed

Some rules such as use of asterisk in SELECT can't be fixed automatically



```
test SQL fluff.sql 1 ●
▶ Run on active connection | ≡ Select block
1 SELECT
2   *,
3   table_name,
4   column_name,
5   table_schema
6 FROM information_schema.tables
7 WHERE column_name = 'id'
8
```

You can check your entire project using the terminal

Ensure your env. is activated. If not, use **conda activate envname**

Use the **sqlfluff lint** command to check the project

```
⊗ (sqlfluff) PS C:\Users\ [redacted] \SQL\SQLFluff> sqlfluff lint --dialect tsq
== [C:\Users\ [redacted] \SQL\SQLFluff\test SQL fluff 2.sql] FAIL
L:   1 | P:   1 | AM04 | Query produces an unknown number of result columns.
      | [ambiguous.column_count]
L:   1 | P:   1 | LT09 | Select targets should be on a new line unless there is
      | only one select target.
```

Each file is checked against all rules in turn

```
L:   5 | P:  30 | LT01 | Unnecessary trailing whitespace at end of file.
      | [layout.spacing]
L:   5 | P:  30 | LT12 | Files must end with a single trailing newline.
      | [layout.end_of_file]
== [C:\Users\ [redacted] \SQL\SQLFluff\test SQL fluff.sql] FAIL
L:   1 | P:   1 | AM04 | Query produces an unknown number of result columns.
      | [ambiguous.column_count]
L:   1 | P:   1 | LT09 | Select targets should be on a new line unless there is
```



sqlfluff fix will attempt to fix all problems in the project

Use .sqlfluff to configure the rules for your project

```
test SQL fluff.sql 9+ ● .sqlfluff × test SQL fluff 2.s
1  # Keywords must be upper case
2  [sqlfluff:rules:capitalisation.keywords]
3  capitalisation_policy = upper
4
5  # Functions must be upper case
6  [sqlfluff:rules:capitalisation.functions]
7  extended_capitalisation_policy = upper
8
9  # Tables must be aliased
10 [sqlfluff:rules:aliasing.table]
11 aliasing = explicit
12
13 # Columns must be aliased
14 [sqlfluff:rules:aliasing.column]
15 aliasing = explicit
16
17 # Aliases must be at least 3 characters long
18 [sqlfluff:rules:aliasing.length]
19 min_alias_length = 3
```

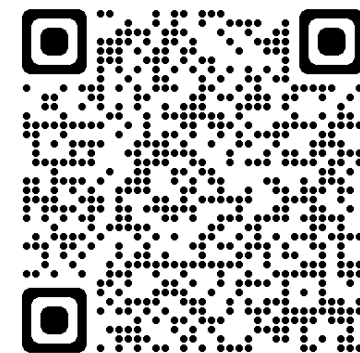
Keywords must be upper case. sqlfluff(CP01)

[View Documentation for Rule CP01.](#)

[View Problem \(Alt+F8\)](#) [Quick Fix... \(Ctrl+.\)](#)

```
from information_schema.tables
WHERE [column_name] = 'id'
```

<https://docs.sqlfluff.com/en/stable/configuration.html>





Takeaways:

1. SQLFluff is used to enforce SQL style and formatting rules
2. We can lint a single query or an entire project (or anywhere between)
3. We can configure which rules are used on a per-project basis



Hi!

I'm Owen and I want to help you flex your data! 🙌

I have 20 years' experience solving tricky data problems.

I have C-suite experience leading global data and programming teams for data products and a host of data-focused technologies in my toolkit (SQL, Python, Advanced Excel, R, Tableau, Power BI, Athena, Glue, Spark, RDS... to name a few).

Let's connect!

LinkedIn: [linkedin.com/in/owenhprice](https://www.linkedin.com/in/owenhprice)

YouTube: [@flexyourdata](https://www.youtube.com/@flexyourdata)

Instagram: [@flexyourdata](https://www.instagram.com/flexyourdata)

Blog: flexyourdata.com