How-To Hours February 23, 2021

dbt-expectations: extending dbt test and the anatomy of a test

dbt_

great_expectations

Claus Herther

@calogica.com

Data & Analytics Consulting

Calogica

dbt-expectations: what is it?

- dbt package with > 50 pre-built dbt schema tests, applicable to models or columns within models
- Port(ish) of the Great Expectations library from Python to SQL/Jinja
- Uses "expectation" semantic to express an assertion

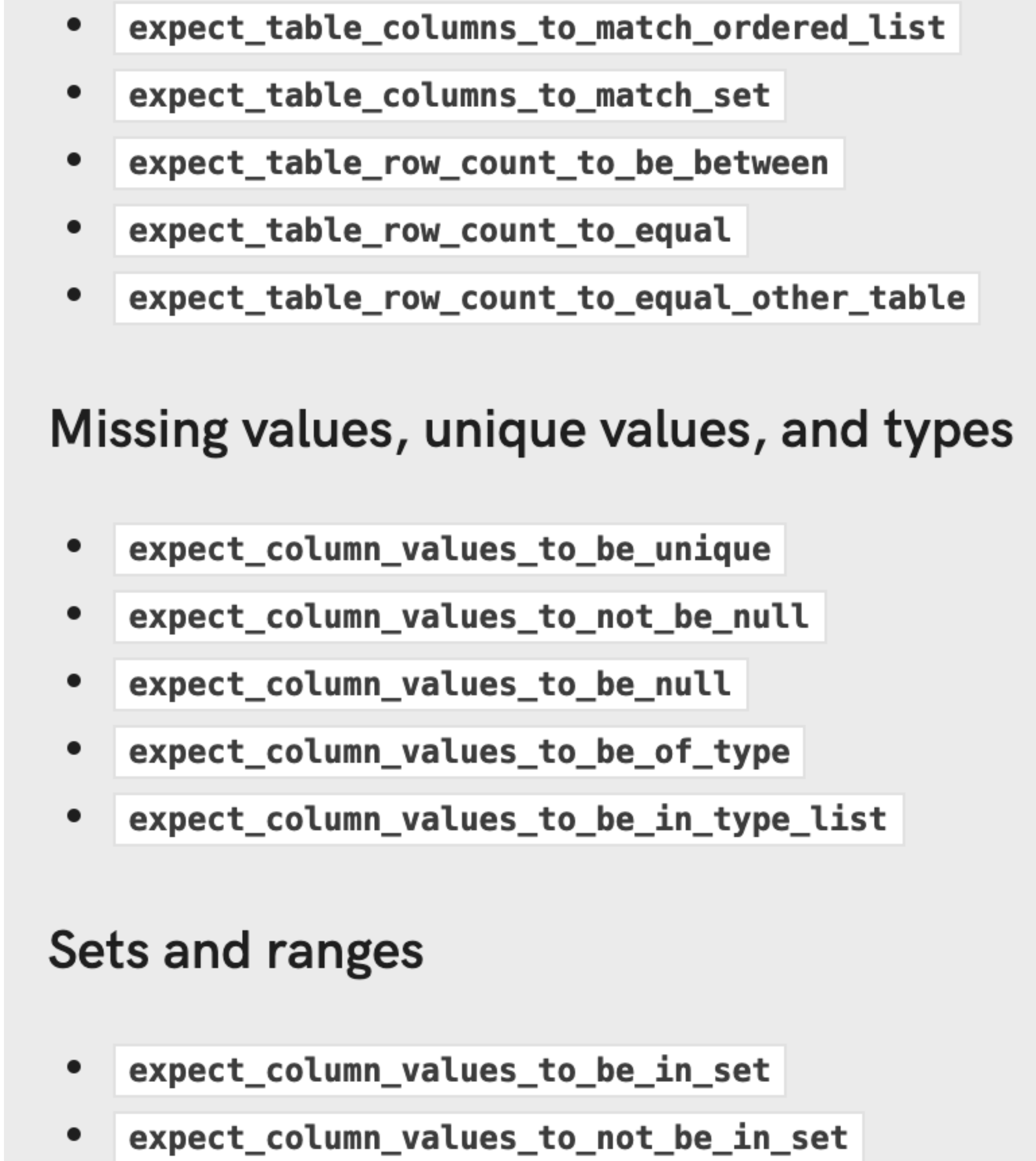
dbt-expectations: what is it?

For example:

```
models: # or seeds or source:
- name: my_model
    tests:
    - dbt_expectations.expect_table_row_count_to_be_between:
        min_value: 1
        max_value: 4
```

dbt-expectations: what is it?

- API is *fairly* close to original GE API (Glossary of Expectations)
- But is "expected" (punny!) to be extensible beyond original API



- `expect_table_columns_to_match_ordered_list`
- `expect_table_columns_to_match_set`
- `expect_table_row_count_to_be_between`
- `expect_table_row_count_to_equal`
- `expect_table_row_count_to_equal_other_table`

Missing values, unique values, and types

- `expect_column_values_to_be_unique`
- `expect_column_values_to_not_be_null`
- `expect_column_values_to_be_null`
- `expect_column_values_to_be_of_type`
- `expect_column_values_to_be_in_type_list`

Sets and ranges

- `expect_column_values_to_be_in_set`
- `expect_column_values_to_not_be_in_set`

dbt-expectations: but why?

- Expectations = Asserts = Tests
- Testing is **native** to dbt
- Most practical tests can be expressed in SQL
- Testing in dbt already fits into your development workflow

dbt-expectations: but why?

- Requires no Python, no additional workflow steps
- Works with dbt Cloud!
- Accomplishes 90%* of what Python would do for you

*wild guess

dbt-expectations: why not?

Use **Great Expectations** when, for example:

- You have a Python-based data-science workflow you need to test
- You need to use expectations that can't be expressed in SQL

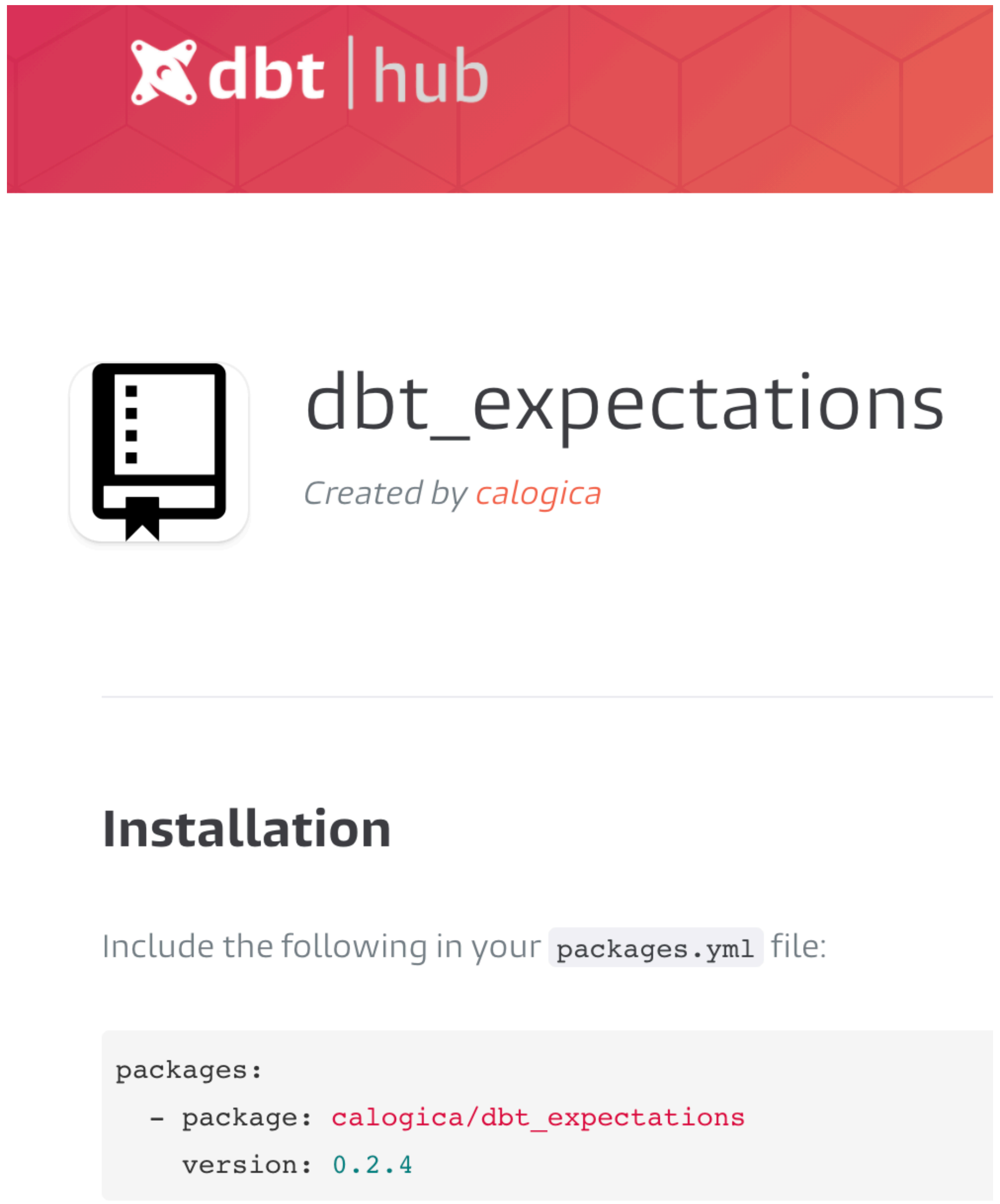
dbt-expectations: how's it work?

Include in `packages.yml`

```yaml
packages:
- package: calogica/dbt_expectations
  version: [">=0.2.0", "<0.3.0"]
```

(for the latest version tag, check:

https://github.com/calogica/dbt-expectations/releases/latest)



dbt_expectations

Created by calogica

Installation

Include the following in your `packages.yml` file:

```yaml
packages:
  - package: calogica/dbt_expectations
    version: 0.2.4
```

dbt-expectations: how's it work?

Apply to models/sources/seeds or columns in `<schema>.yml`

```
models:
- name: my_model
    tests:
    - dbt_expectations.expect_table_row_count_to_equal:
        value: 4

    columns:
    - name: my_column
        tests:
        - dbt_expectations.expect_column_values_to_be_between:
            min_value: 0
            max_value: 10
```

dbt-expectations: how's it work?

Most expectations are simple expressions passed to `expression_is_true` macro

```
{% macro test_expect_column_values_to_not_be_null(model, column_name, row_condition=None) %}

{% set expression = column_name ~ " is not null" %}

{{ dbt_expectations.expression_is_true(model,
                                      expression=expression,
                                      row_condition=row_condition
                                      )
                                      }}
{% endmacro %}
```

dbt-expectations: how's it work?

A simple expression evaluator, e.g. `dbt_utils.expression_is_true`

```
{% macro test_expression_is_true(model, condition='true') %}
{% set expression = kwargs.get('expression', kwargs.get('arg')) %}
with meet_condition as (
    select * from {{ model }} where {{ condition }}
),
validation_errors as (
    select
        *
    from meet_condition
    where not({{expression}})
)
select count(*) from validation_errors
{% endmacro %}
```

dbt-expectations: wait, you mean I can write my own schema tests?

Yes! And you should. Schema tests are "just" macros!

```
{% macro test_my_custom_validation(model,
                                    column_name,
                                    bad_value='bad value') %}
with validation_errors as (

    select
        count(*)
    from
        {{ model }}
    where {{ column_name }} = '{{ bad_value }}'

)
select count(*)
from validation_errors

{% endmacro %}
```

dbt-expectations: wait, you mean I can write my own schema tests?

Or, using `dbt_utils.expression_is_true`

```
{% macro test_my_custom_validation(model, column_name, bad_value='bad value') %}

{% set expression = column_name ~ " != '" ~ bad_value ~ "'" %}

{{ dbt_utils.expression_is_true(model, expression=expression) }}

{% endmacro %}
```

dbt-expectations: wait, you mean I can write my own schema tests?

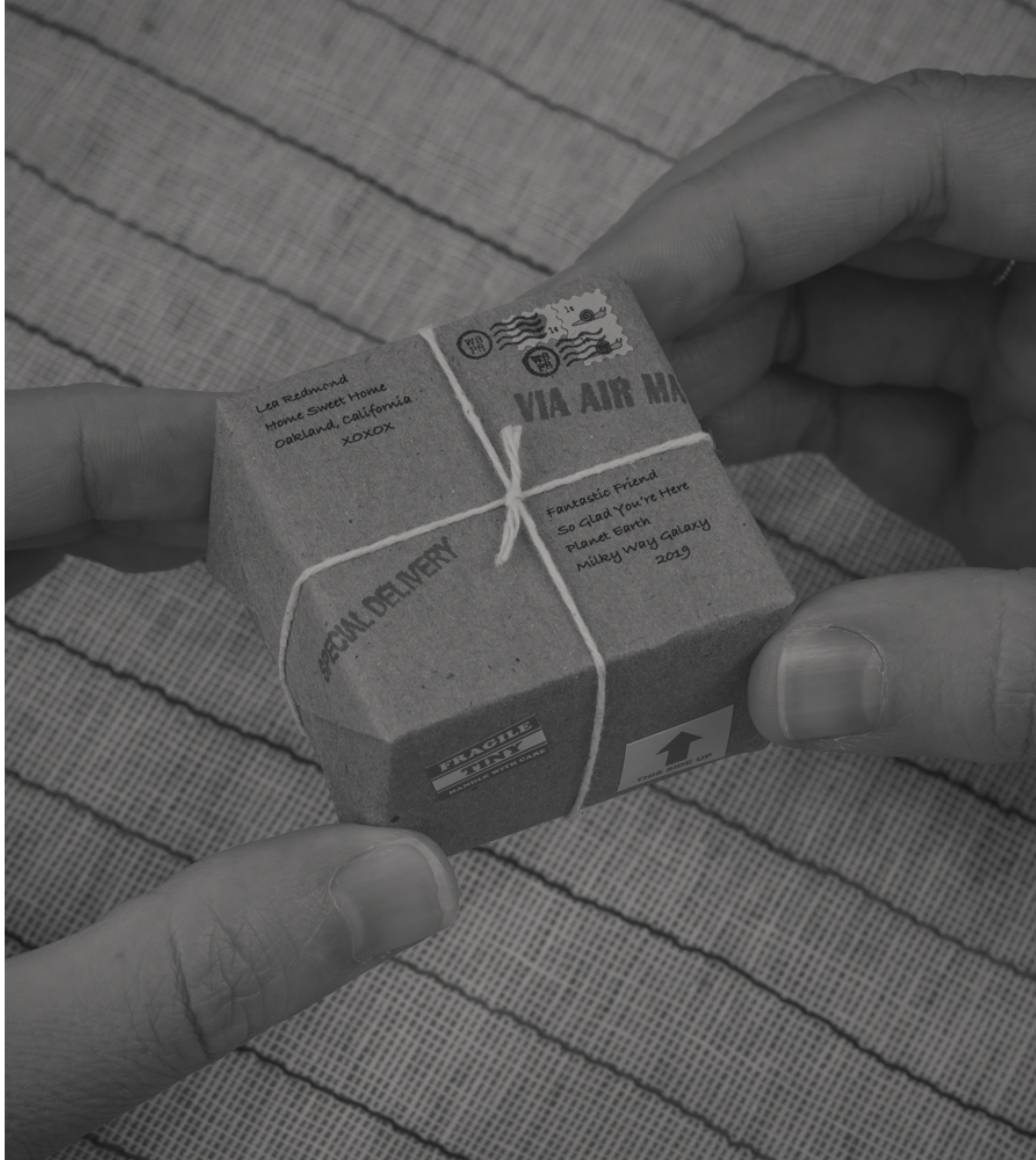
```
models:
  - name: my_model
    columns:
    - name: my_column
      tests:
        - my_custom_validation:
            bad_value: 'super bad value'
```

dbt-expectations: so like, what's a dbt package then?

- It's a separate project that can contain models and/or macros
- You import it via `packages.yml`

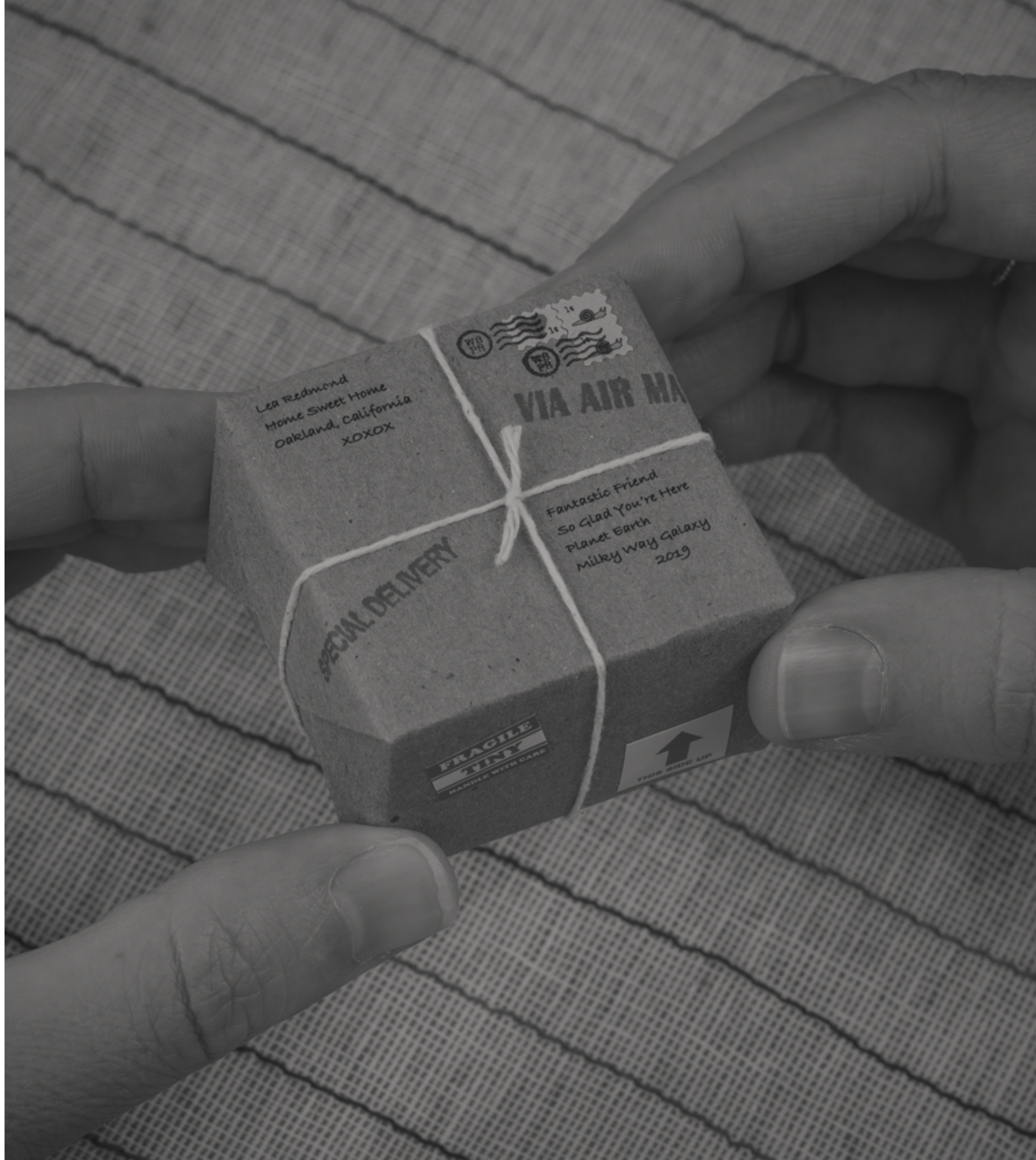
dbt-expectations: how do I create my own packages?

- Create a separate project that contains models and/or macros you'd like to share
- Publish on dbt hub
- Profit! 💰



dbt-expectations: how do I create my own packages?

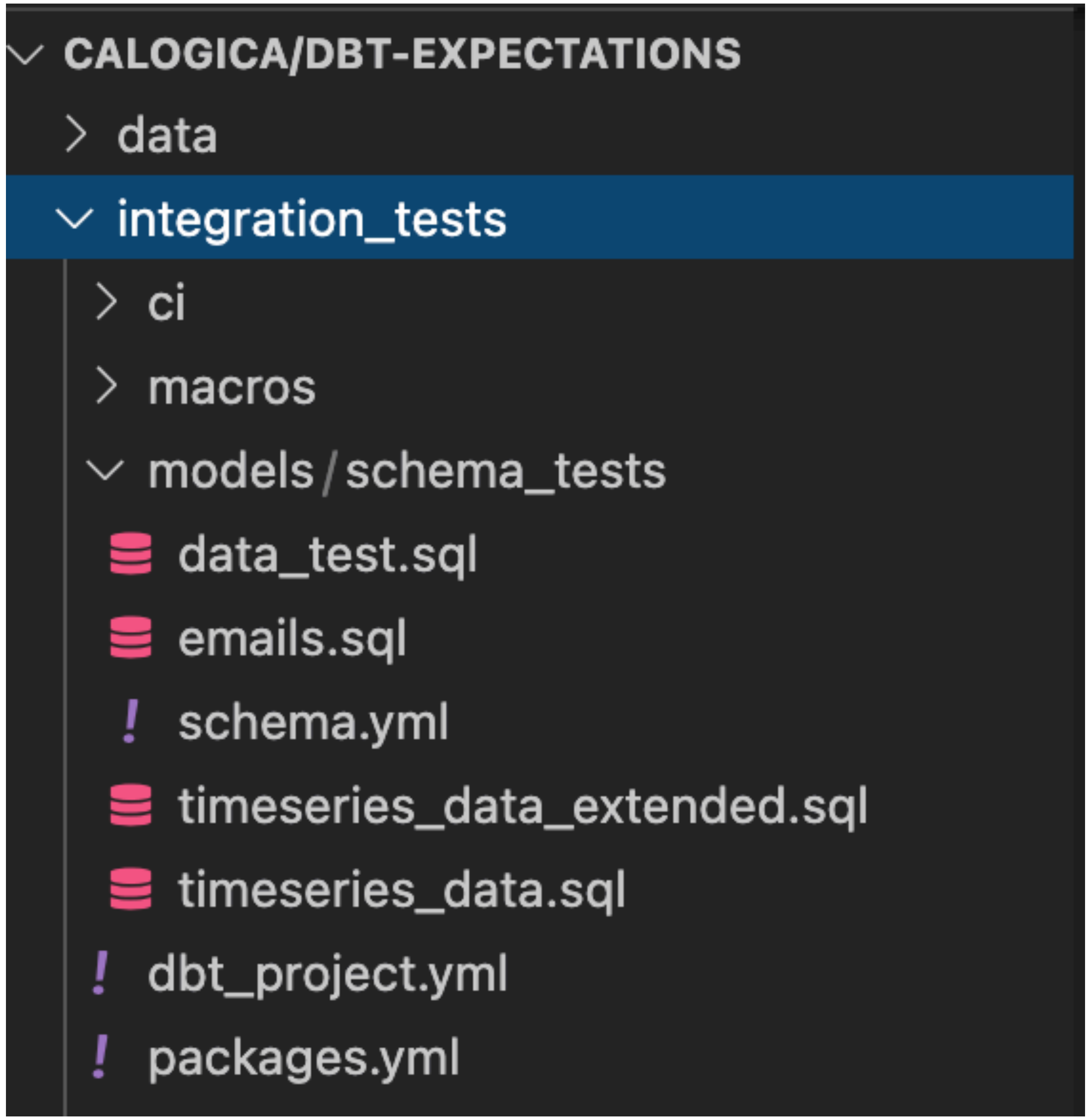
- BUT: You need to **test** your packages!



dbt-expectations: how do I test my package?

- Create an `integration_tests` project within your package project
- Privately import your package

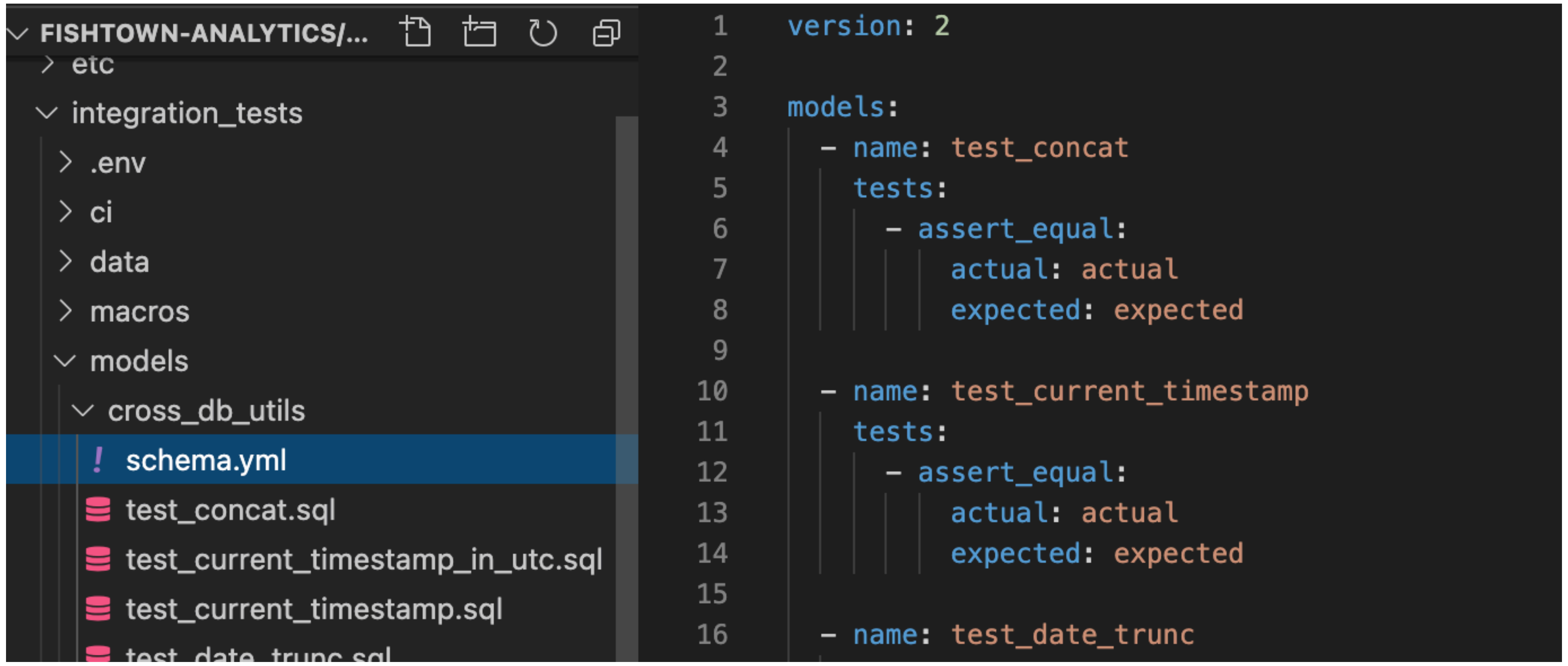
```
packages:
- local: ../
```



```
CALOGICA/DBT-EXPECTATIONS
  > data
  v integration_tests
    > ci
    > macros
    v models / schema_tests
      data_test.sql
      emails.sql
      schema.yml
      timeseries_data_extended.sql
      timeseries_data.sql
    dbt_project.yml
    packages.yml
```

dbt-expectations: how do I test my package?

- Write test models that use your macros and test those:



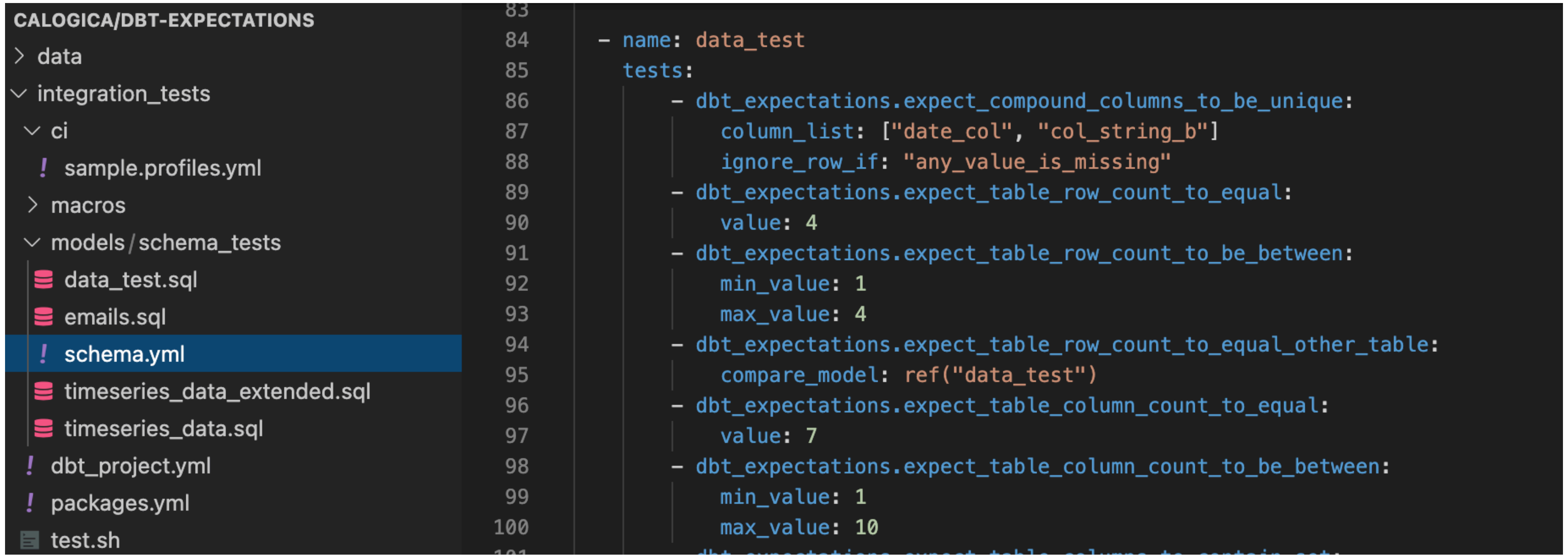
FISHTOWN-ANALYTICS/...

```
> etc
v integration_tests
  > .env
  > ci
  > data
  > macros
  v models
    v cross_db_utils
      schema.yml
      test_concat.sql
      test_current_timestamp_in_utc.sql
      test_current_timestamp.sql
```

```yaml
version: 2

models:
  - name: test_concat
    tests:
      - assert_equal:
          actual: actual
          expected: expected

  - name: test_current_timestamp
    tests:
      - assert_equal:
          actual: actual
          expected: expected

  - name: test_date_trunc
```

dbt-expectations: how do I test my package?

- Write test models and schema files that use your test macros

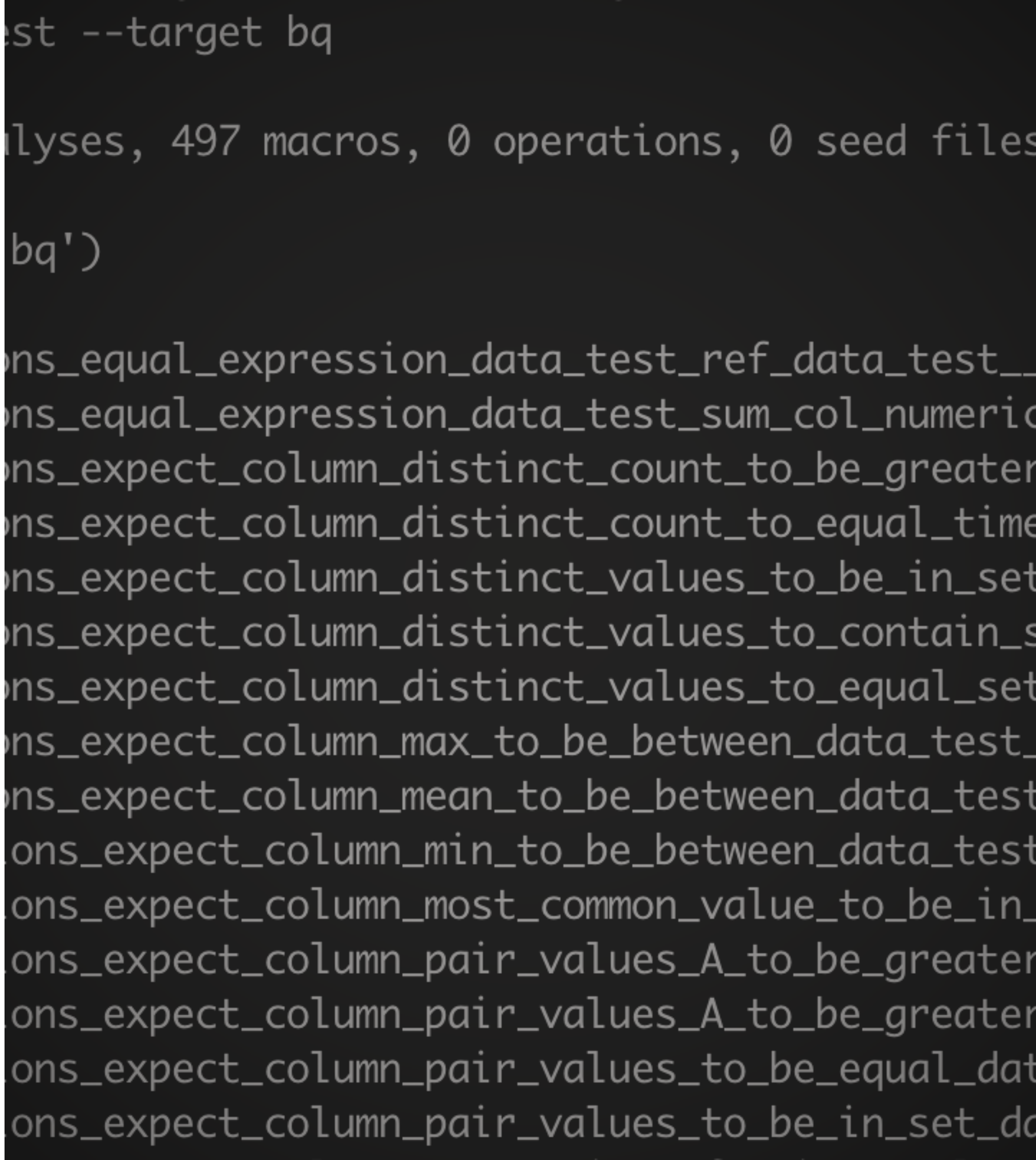


```
CALOGICA/DBT-EXPECTATIONS
> data
v integration_tests
  v ci
    ! sample.profiles.yml
  > macros
  v models / schema_tests
    data_test.sql
    emails.sql
    ! schema.yml
    timeseries_data_extended.sql
    timeseries_data.sql
  ! dbt_project.yml
  ! packages.yml
  test.sh
```

```yaml
  - name: data_test
    tests:
        - dbt_expectations.expect_compound_columns_to_be_unique:
            column_list: ["date_col", "col_string_b"]
            ignore_row_if: "any_value_is_missing"
        - dbt_expectations.expect_table_row_count_to_equal:
            value: 4
        - dbt_expectations.expect_table_row_count_to_be_between:
            min_value: 1
            max_value: 4
        - dbt_expectations.expect_table_row_count_to_equal_other_table:
            compare_model: ref("data_test")
        - dbt_expectations.expect_table_column_count_to_equal:
            value: 7
        - dbt_expectations.expect_table_column_count_to_be_between:
            min_value: 1
            max_value: 10
```

dbt-expectations: how do I test my package?

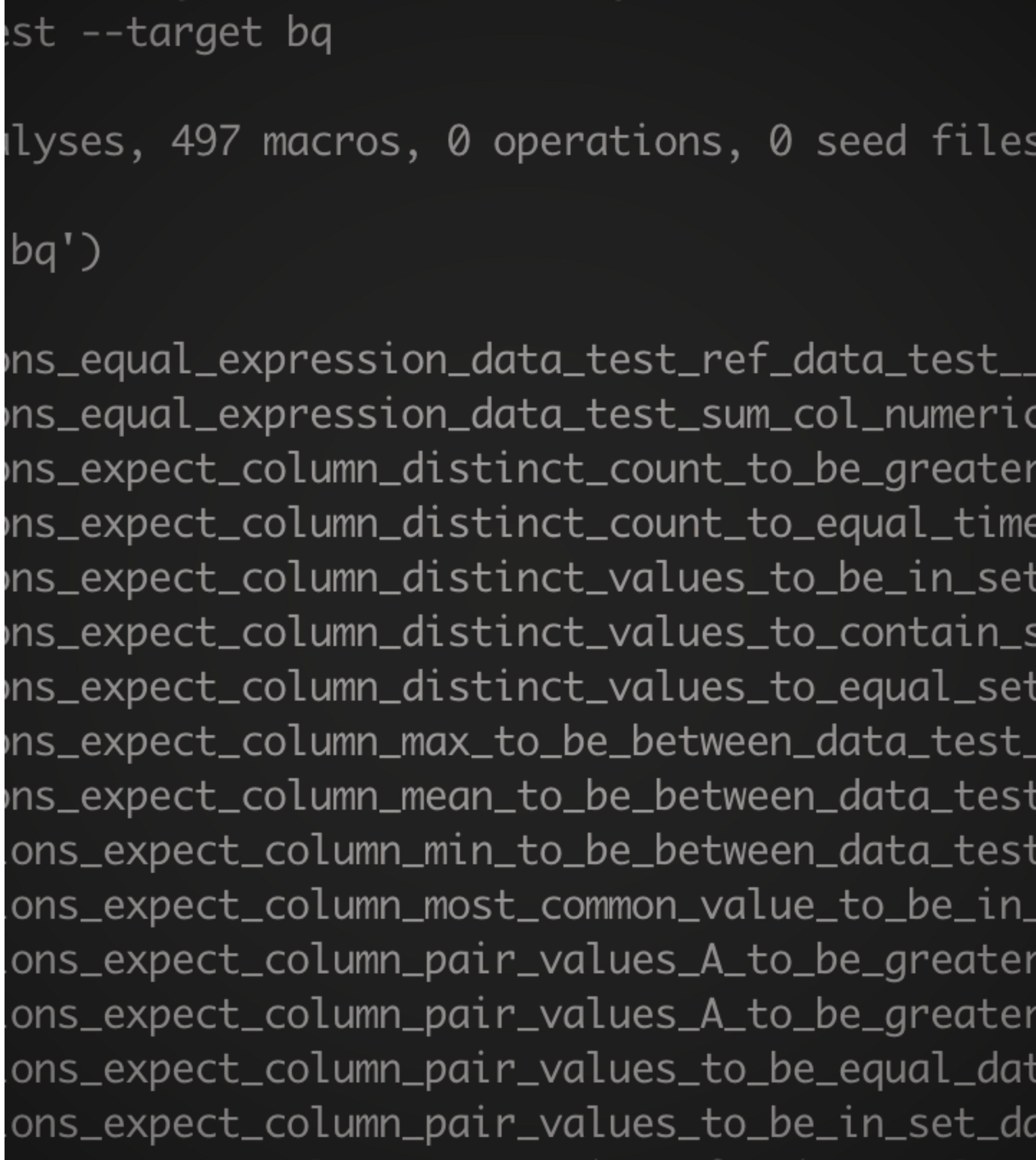
- Try to test on as many platforms as possible (the ones you support)
- Use `dbt-utils` or your own adapter macros to make cross-platform support easier



```
est --target bq

lyses, 497 macros, 0 operations, 0 seed files

bq')

ons_equal_expression_data_test_ref_data_test_
ons_equal_expression_data_test_sum_col_numeric
ons_expect_column_distinct_count_to_be_greater
ons_expect_column_distinct_count_to_equal_time
ons_expect_column_distinct_values_to_be_in_set
ons_expect_column_distinct_values_to_contain_s
ons_expect_column_distinct_values_to_equal_set
ons_expect_column_max_to_be_between_data_test_
ons_expect_column_mean_to_be_between_data_test
ons_expect_column_min_to_be_between_data_test
ons_expect_column_most_common_value_to_be_in_
ons_expect_column_pair_values_A_to_be_greater
ons_expect_column_pair_values_A_to_be_greater
ons_expect_column_pair_values_to_be_equal_dat
ons_expect_column_pair_values_to_be_in_set_da
```

dbt-expectations: how do I test my package?

- Make sure your tests pass before submitting a PR and before committing to `main`!



```
est --target bq

lyses, 497 macros, 0 operations, 0 seed files

bq')

ons_equal_expression_data_test_ref_data_test_
ons_equal_expression_data_test_sum_col_numeric
ons_expect_column_distinct_count_to_be_greater
ons_expect_column_distinct_count_to_equal_time
ons_expect_column_distinct_values_to_be_in_set
ons_expect_column_distinct_values_to_contain_s
ons_expect_column_distinct_values_to_equal_set
ons_expect_column_max_to_be_between_data_test_
ons_expect_column_mean_to_be_between_data_test
ons_expect_column_min_to_be_between_data_test
ons_expect_column_most_common_value_to_be_in_
ons_expect_column_pair_values_A_to_be_greater
ons_expect_column_pair_values_A_to_be_greater
ons_expect_column_pair_values_to_be_equal_dat
ons_expect_column_pair_values_to_be_in_set_da
```

dbt-expectations: how can you contribute?

- File an issue
- Submit a PR
- Improve the docs

Questions?

claus@calogica.com