10. Macros

Introduction to Macros

Macros in DBT are reusable pieces of SQL code, similar to functions in other programming languages. They use Jinja, a templating language, to generate SQL dynamically.

Benefits of macros:

- Reduce code duplication
- Encapsulate complex logic
- Make your code more maintainable

Creating Custom Macros

To create a macro, add a file to the `macros` directory:

`macros/generate_schema_name.sql`:

sql

```
{% macro generate_schema_name(custom_schema_name, node) %}

    {%- set default_schema = target.schema -%}

    {%- if custom_schema_name is none -%}
        {{ default_schema }}
    {%- else -%}
        {{ default_schema }}_{{ custom_schema_name }}
    {%- endif -%}

{% endmacro %}
```

This macro determines the schema name based on the target schema and an optional custom schema.

Using Built-in Macros

DBT comes with several built-in macros that you can use in your models:

1. ref()

The most common macro, used to reference other models:

sql

```
SELECT * FROM {{ ref('stg_customers') }}
```

2. source()

References raw data sources:

sql

```
SELECT * FROM {{ source('raw', 'customers') }}
```

3. config()

Sets model configuration:

sql

```
{{ config(
    materialized='table',
    schema='marketing'
) }}
```

4. target

References information about the current target:

sql

```
{% if target.name == 'prod' %}
    SELECT * FROM production_schema.table
{% else %}
```

```
    SELECT * FROM development_schema.table
{% endif %}
```

Creating Complex Macros

Macros can include logic, loops, and conditionals:

sql

```
{% macro column_list(columns, prefix=none) %}
    {% for column in columns %}
        {% if prefix %}{{ prefix }}.{% endif %}{{ column }}{% if not
loop.last %},{% endif %}
    {% endfor %}
{% endmacro %}

-- Usage:
SELECT {{ column_list(['id', 'name', 'email']) }} FROM customers
-- Generates: SELECT id, name, email FROM customers
```

Macros can also call other macros:

sql

```
{% macro star(from, relation_alias=none, except=[]) %}

    {%- set include_cols = [] -%}

    {%- set columns = adapter.get_columns_in_relation(from) -%}
    {%- for column in columns -%}
        {%- if column.name not in except -%}
            {%- do include_cols.append(column.name) -%}
        {%- endif %}
    {%- endfor %}

    {{ column_list(include_cols, relation_alias) }}

{% endmacro %}

-- Usage:
SELECT {{ star(ref('customers'), except=['sensitive_column']) }} FROM {{
ref('customers') }}
```