dbt ❤️ Jinja Cheatsheet

Variables

```
{% set my_string = "example" %}
{% set my_list = ["apple", "lemon"] %}
{% set my_dict = {"WatchEvent": "watch_user_count",
                  "ForkEvent": "fork_user_count" %}
```

Comments

```
{# Example comment #}
```

Statements

```
{% … %} e.g.: loops, if
```

Expressions

```
{{ … }}  e.g.: ref
```

var()

```
SELECT
  *
FROM events
WHERE event_type = '{{ var("event_type") }}'
```

Macros

Macros are reusable code snippets (functions) written in Jinja

```
{% macro cents_to_dollar(col_name, precision=2) %}
({{ col_name }} / 100)::numeric(16, {{ precision }})

{% endmacro %}
```

Run macro

```
dbt run-operation <macro> --args '{example: value}'
```

Loops

Input

```
-- Over list
{% set my_list = ['sales_x', 'sales_y'] %}
SELECT
  id,
  {%- for col_name in my_list %}
  SUM({{ col_name }})
  {%- if not loop.last -%}, {%- endif -%}
  {% endfor %}
FROM example
GROUP BY 1
```

```
-- Over dictionary
{% set payment_methods = {"type_0" : "bank_transfer",
                          "type_1" : "credit_card",
                          "type_2" : "gift_card"} %}

SELECT
  order_id,
  {%- for type, column_name in payment_methods.items()%}
        sum(CASE
              WHEN payment_method = '{{type}}'
              THEN amount end) as {{ column_name }}_amt
  {%- if not loop.last -%}, {%- endif -%}
  {%- endfor -%}
FROM example
GROUP BY 1
```

Compiled

```
SELECT
  id,
  SUM(sales_x),
  SUM(sales_y)
FROM example
GROUP BY 1
```

```
SELECT
  order_id,
  sum(CASE
        WHEN payment_method = 'type_0'
        THEN amount END) AS bank_transfer_amt,
  sum(CASE
        WHEN payment_method = 'type_1'
        THEN amount END) AS credit_card_amt,
  sum(CASE
        WHEN payment_method = 'type_2'
        THEN amount END) AS gift_card_amt
FROM example
GROUP BY 1
```

Graph (dag)

```
{% macro example() %}

{% if execute %}
{% for node in graph.nodes.values() %}
{% do log(node.unique_id ~ ", config: " ~ node.config,
info=true) %}
{% endfor %}
{% endif %}

{% endmacro %}
```

dbt_utils

dbt_utils is a collection of reusable dbt macros
Examples:

deduplicate - remove duplicates from a model
group_by - build a group by statement for (1..N)

Cast variable

```
# To string
{% set my_int_var = 2020 %}
{% set my_int_var|string %}

# To int
{% set my_str_var = "2020" %}
{% set my_str_var|int %}
```

if |elif|else

```
{% macro generate_schema_name() -%}
{%- if target.name == 'dev'-%}
{%- elif target.name == 'prod' -%}
{%- else -%}
{%- endif -%}
{%- endmacro %}
```

Trim whitespace

`{%- … %}` Strips before

`{%- … -%}` Strips before and after

Length

```
{% set my_list = ["apple", "lemon"] %}

# Check if my_list has more than 3 element
{% if products|length > 3 %}
```

Exceptions

```
-- Warning
{% do exceptions.warn("Warning message") %}

-- Error
{{ exceptions.raise_compiler_error("Error message") }}
```

Logging to Stdout

```
{{ log("Some text" ~ my_string, info=True) }}
```

Manipulate objects

```
{% set numbers = [] %}

{%- for i in range(1, 10) %}
{%- do numbers.append(i) -%}

{%- endfor %}
```

Debug

The {{ debug() }} macro will open an iPython debugger in the context of a compiled dbt macro

Usage:

```
...
{{ debug() }}
...
```

Return

```
{% macro example() %}

{{ return("Hello") }}

{% endmacro %}
```

Enviroment variables

```
{{ env_var("VAR_NAME") }}
```

Loop.last

To avoid trailing commas in loops use:

```
{% if not loop.last %},{% endif %}
```

Run query

```
{% set results = run_query("select * from table") %}
{% do results.print_table() %}
```

Print

```
{{ print(("My Message") }}
```