Introducing Jinja with dbt

CASE STUDY: BUILDING E-COMMERCE DATA MODELS WITH DBT

Susan Sun
Freelance Data Scientist

datacamp

dbt uses three languages

dbt uses **three** languages:

- **SQL** for transformations
- **YAML** for documenting and testing
- **Jinja** for templating

- Example of SQL:

```
SELECT * FROM ...
```

- Example of YAML:

```
- name: customers
  description: A data mart
```

- Example of Jinja:

```
FROM {{ ref('stg_looker__users') }}
```

What is dbt Jinja

What is dbt Jinja?

- A Pythonic templating language
- Used to enable dynamic SQL generation
- Found in dbt models, macros, tests, etc.

Why is it useful?

- Better collaboration
- Less repetition (DRY), more reusable code

Examples of Jinja in dbt:

- Referencing dbt models

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

- Creating loops

```
{% for column in ['col1', 'col2'] %}
  SELECT {{ column }}
  FROM table_name
{% endfor %}
```

Types of dbt Jinja

1. Jinja **statements** are enclosed in `{%...%}`

```
{% set order_statuses = ['Shipped', 'Complete', 'Processing'] %}
```

2. Jinja **expressions** are enclosed in `{{...}}`

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

3. Jinja **comments** are enclosed in `{# #}`

```
{# This is a comment #}
```

Jinja statements: set

- Types of Jinja statements: **set**, **loop**, **conditional** (if/else), **macros**, etc.
- A **set** Jinja statement creates a variable and assigns value.
  - Template:

```
{% set ... %}
```

  - Example:

```
{% set country = 'Australia' %}
```

Jinja and dbt compile

Usage

- Inside `customers.sql`, as written:

```
{% set country = 'Australia' %}

SELECT ...
FROM ...
WHERE country = '{{ country }}'
```

Compiled code

- `dbt compile -s customers.sql` output:

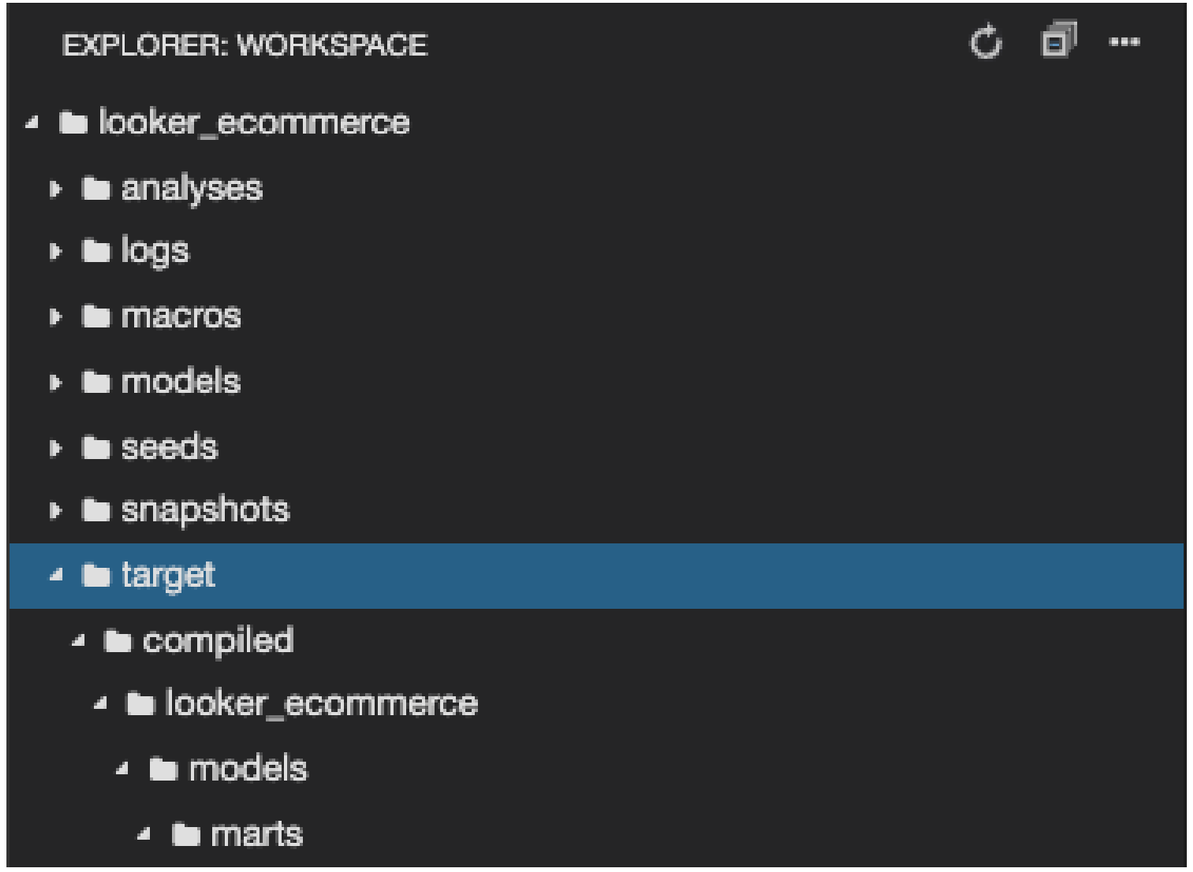
```
SELECT ...
FROM ...
WHERE country = 'Australia'
```

Jinja and dbt compile

- Terminal output for a single model
- `dbt compile -s customers.sql` output:

```
Running with dbt=1.8.4
...
Compiled node 'customers' is:
WITH customer_base AS (
    SELECT
...
    FROM "dbt"."main"."stg_looker__users"
    WHERE country = 'Australia'
)
...
```

- Generated `/target/compiled` directory
- `dbt compile` generated compiled files:



Let's practice!

CASE STUDY: BUILDING E-COMMERCE DATA MODELS WITH DBT

Complex usage of jinja with dbt

CASE STUDY: BUILDING E-COMMERCE DATA MODELS WITH DBT

Susan Sun
Freelance Data Scientist

datacamp

Jinja statement: loop

Jinja **statements** are enclosed in `{% %}` :

- **Looping** through items
  - Template:

```
{% for ... %} ... {% endfor %}
```

  - Example:

```
{% for order_status in order_statuses %}
  SUM(
      CASE WHEN status = '{{ order_status }}'
        THEN order_id
      END)
{% endfor %}
```

Jinja statement: loop

Repeating SQL:

```
SELECT
    user_id,
    SUM(CASE WHEN status = 'Shipped' THEN 1 ELSE 0 END)
        AS num_orders_Shipped,
    SUM(CASE WHEN status = 'Complete' THEN 1 ELSE 0 END)
        AS num_orders_Complete,
    SUM(CASE WHEN status = 'Processing' THEN 1 ELSE 0 END)
        AS num_orders_Processing,
FROM {{ ref('stg_looker__orders') }}
GROUP BY 1
```

Jinja statement: loop

```
{% set order_statuses = ['Shipped', 'Complete', 'Processing'] %}

SELECT
    user_id,
    -- Jinja loop
    {% for order_status in order_statuses %}
        SUM(CASE WHEN status = '{{ order_status }}' THEN 1 ELSE 0 END)
        -- Parametrized column name
        AS num_orders_{{ order_status }}
    {% endfor %}
FROM {{ ref('stg_looker__orders') }}
GROUP BY 1
```

Jinja statement: loop

`dbt compile` output:

```
SELECT
    user_id,
    SUM(CASE WHEN status = 'Shipped' THEN 1 ELSE 0 END)
        AS num_orders_Shipped,
    SUM(CASE WHEN status = 'Complete' THEN 1 ELSE 0 END)
        AS num_orders_Complete,
    SUM(CASE WHEN status = 'Processing' THEN 1 ELSE 0 END)
        AS num_orders_Processing,
FROM "dbt"."main"."stg_looker__orders"
GROUP BY 1
```

Jinja statement: macros

A Jinja macro:

- Is a function, written in Jinja
- Is reusable
- Is stored in the `/macros` folder

`COALESCE()`:

- returns the first non-null value
- syntax: `COALESCE(value1, value2, ...)`
- e.g. `COALESCE(nickname, first_name)`

An example of a macro function:

coalesce_and_round.sql

```
{%- macro coalesce_and_round(column_name, decimal_places = 2) -%}
    ROUND(COALESCE({{ column_name }}, 0), {{ decimal_places }})
{%- endmacro %}
```

Jinja statement: macros

Jinja **statements** are enclosed in `{% %}` :

- A reusable **macro** function:
  - Template:

```
{% macro ... %} ... {% endmacro %}
```

  - Example:

```
{% macro coalesce_and_round(column_name, decimal_places = 2) %}
      ROUND(COALESCE({{ column_name }}, 0), {{ decimal_places }})
{% endmacro %}
```

Jinja statement: macros

- Repeated SQL

```
ROUND(COALESCE(sales_amount, 0), 2) AS sales_amount,
ROUND(COALESCE(cost_of_goods_sold, 0), 2) AS cost_of_goods_sold,
```

- DRY (**D**on't **R**epeat **Y**ourself)
  - Create macro called `coalesce_and_round.sql`
  - Store in `/macros`
  - Invoke:

```
{{ coalesce_and_round('sales_amount', 2) }} AS sales_amount,
{{ coalesce_and_round('cost_of_goods_sold', 2) }} AS cost_of_goods_sol
```

Managing Jinja whitespace

As written:

```
{% set traffic_source_values = ['Adwords', 'Email', 'Facebook'] %}
{% set browser_values = ['Chrome', 'Firefox', 'Safari', 'IE', 'Other'] %}

SELECT
    user_id,
    COUNT(DISTINCT session_id) AS num_web_sessions,
...
```

Managing Jinja whitespace

Compiled:

```
SELECT
    user_id,
    COUNT(DISTINCT session_id) AS num_web_sessions,
...
```

Managing Jinja whitespace

- `{%- ... %}` strips the whitespace **before**.
- `{% ... -%}` strips the whitespace **after**.
- `{%- ... -%}` strips **both**.
- An example:

```
{%- set traffic_source_values = ['Adwords', 'Email', 'Facebook'] -%}
{%- set browser_values = ['Chrome', 'Firefox', 'Safari', 'IE', 'Other'] -%}

SELECT
    user_id,
    COUNT(DISTINCT session_id) AS num_web_sessions,
...
```

Let's practice!

CASE STUDY: BUILDING E-COMMERCE DATA MODELS WITH DBT

Recap: dbt case study

CASE STUDY: BUILDING E-COMMERCE DATA MODELS WITH DBT

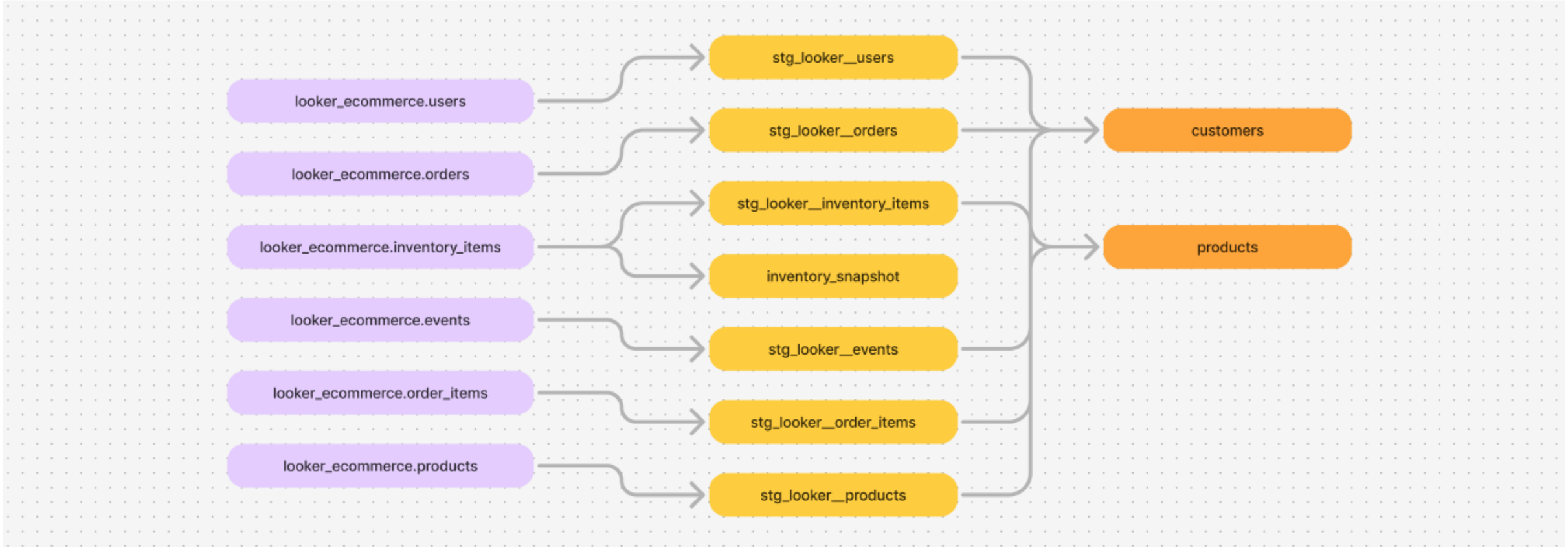
Susan Sun
Freelance Data Scientist

Chapter 1: Setting up dbt

1. Set up:
   - Check installtion: `dbt --versions`, `which dbt`
   - Initialize dbt: `dbt init`
2. Load data:
   - Load dbt seed files (e.g. distribution center): `dbt seed`
   - Load dbt source files. E.g. `{{ source('looker_ecommerce', 'orders') }}`
3. Review dbt subcommands: `dbt run`, `dbt test`, `dbt build`
4. Review dbt project file and folder structure

Chapter 2: Building dbt models

1. Build dbt staging, mart, and snapshot models
2. Add table/column documentation and dbt data tests



Chapter 3: Jinja with dbt

1. Reduce repeated code using Jinja expressions: **set**, **loop**, **macros**
2. Use dbt compile and whitespace management for development

```
{%- set order_statuses = ['Shipped', 'Complete', 'Processing'] -%}

SELECT
    user_id,
    {%- for order_status in order_statuses %}
        COUNT(DISTINCT CASE WHEN status = '{{ order_status }}' THEN order_id END)
        AS num_orders_{{ order_status }}
    {%- endfor %}
FROM {{ ref('stg_looker__orders') }}
GROUP BY 1
```

Congratulations!

CASE STUDY: BUILDING E-COMMERCE DATA MODELS WITH DBT