**1. Sources in dbt Source.yml file**

* What it is:
  + A way to define raw data tables from your warehouse (like Snowflake, BigQuery, Redshift, etc.) that dbt will use as inputs.
  + Helps track lineage (where your data is coming from).

sources:

       - name: ecom

         schema: raw

         description: E-commerce data for the Jaffle Shop

         tables:

         - name: raw\_customers

           description: One record per person who has purchased one or more items

           columns:

            - name: id

              data\_tests:

                - unique

                - not\_null

            - name: name

              data\_tests:

                - not\_null

         - name: raw\_orders

           description: One record per order (consisting of one or more order items)

           loaded\_at\_field: ordered\_at

         - name: raw\_items

           description: Items included in an order

         - name: raw\_stores

           loaded\_at\_field: opened\_at

         - name: raw\_products

           description: One record per SKU for items sold in stores

         - name: raw\_supplies

           description: One record per supply per SKU of items sold in stores

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

**2. Models in dbt**

* What it is:
  + The core building blocks of dbt.
  + A model is just a SQL file that transforms data into a clean, usable format (staging, marts, reporting).

SELECT

order\_id,

customer\_id,

order\_date,

amount

FROM {{ source('raw', 'orders') }}

WHERE order\_status = 'COMPLETE'

**Macros in dbt**

* What it is:
  + Like functions in programming, written in Jinja (templating language).
  + They help avoid repetitive SQL.

{% macro generate\_schema\_name(custom\_schema\_name, node = none) -%}

    {%- set default\_schema = target.schema -%}

    {%- if custom\_schema\_name is none -%}

        {{ default\_schema }}

    {%- else -%}

        {{ custom\_schema\_name | trim }}

    {%- endif -%}

{%- endmacro %}

SELECT \* FROM orders WHERE {{ get\_last\_n\_days('order\_date', 30) }}

**Snapshots in dbt**

* What it is:
  + Used for slowly changing dimensions (SCD Type 2).
  + Captures historical changes in a table over time

{% snapshot customers\_snapshot %}

{{

config(

target\_schema='snapshots',

unique\_key='customer\_id',

strategy='timestamp',

updated\_at='last\_updated'

)

}}

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

{% endsnapshot %}

**Seeds in dbt**

* What it is:
  + CSV files that dbt loads into your warehouse as tables.
  + Useful for lookup tables, reference data, or test data.
* Example (seeds/country\_codes.csv):

country\_code,country\_name

US,United States

IN,India

UK,United Kingdom

* Run with:

dbt seed

**dbt\_project.yml file**

# Name your project! Project names should contain only lowercase characters

# and underscores. A good package name should reflect your organization's

# name or the intended use of these models

name: 'jaffle\_shop'

version: '1.0.0'

config-version: 2

# This setting configures which "profile" dbt uses for this project.

profile: 'default'

# These configurations specify where dbt should look for different types of files.

# The `model-paths` config, for example, states that models in this project can be

# found in the "models/" directory. You probably won't need to change these!

model-paths: ["models"]

analysis-paths: ["analyses"]

test-paths: ["tests"]

seed-paths: ["seeds"]

macro-paths: ["macros"]

snapshot-paths: ["snapshots"]

target-path: "target"  # directory which will store compiled SQL files

clean-targets:         # directories to be removed by `dbt clean`

  - "target"

  - "dbt\_packages"

vars:

    "dbt\_date:time\_zone": "America/Los\_Angeles"

models:

  +database: dbt\_db\_dev

  +schema: staging

  jaffle\_shop:

    staging:

       +materialized: view

    marts:

      +schema: marts

      +materialized: table

# Configuring models

# Full documentation: https://docs.getdbt.com/docs/configuring-models

# In dbt, the default materialization for a model is a view. This means, when you run

# dbt run or dbt build, all of your models will be built as a view in your data platform.

# The configuration below will override this setting for models in the example folder to

# instead be materialized as tables. Any models you add to the root of the models folder will

# continue to be built as views. These settings can be overridden in the individual model files

# using the `{{ config(...) }}` macro.

models:

  my\_new\_project:

    # Applies to all files under models/example/

    example:

      +materialized: table