HOOKS:

In dbt, hooks are a feature that allow you to execute custom SQL statements or commands at specific points during the execution of your dbt models. Hooks provide a way to perform additional operations before or after the execution of models, such as running setup or cleanup tasks, interacting with external systems, or logging audit information.

Hooks can be defined at different levels in dbt, including at the model level, the project level, or even globally for all projects. When multiple hooks are defined at different levels, the execution sequence depends on the level and order of the hooks.

The sequence of hook execution is as follows:

1. Model-level hooks: Hooks defined directly within a specific model are executed first.
2. Project-level hooks: Hooks defined at the project level, outside any specific model, are executed next. These hooks apply to all models within the project.
3. Global hooks: Hooks defined globally, outside any specific project or model, are executed last. These hooks apply to all projects.

Order on which hooks will be executed:

1. Hooks from dependent packages: If you have hooks defined in dependent packages, those hooks will be executed before hooks in the active package. Dependencies can be specified in your **dbt\_project.yml** file using the **packages** configuration.
2. Hooks defined in **dbt\_project.yml**: Hooks defined in the **dbt\_project.yml** file will be executed after any hooks from dependent packages. These hooks can be defined at the project level and apply to all models within the project.
3. Hooks within the model itself: Hooks defined directly within a specific model will be executed after the hooks defined in the **dbt\_project.yml**. These hooks provide flexibility to customize the behavior of individual models.
4. Hooks within a given context: If you have multiple hooks defined within the same context, such as pre-hooks or post-hooks, they will be executed in the order in which they are defined. This allows you to control the sequence of operations within a specific hook context.

## ON-RUN START / END

Hooks can be defined as on-run start/end in dbt\_project.yml and with it, it will run this sql queries before the model run and after the run has completed in on-run-start and on-run-end types of hooks.

***It is for all the models, when any of the model is run then these hooks will be triggered and act as global hooks.***

# Hook defining:

* on-run-start: ["truncate table dbt\_raw.hooks.dbt\_audits", "Insert into dbt\_raw.hooks.dbt\_audits (audit\_type) values ('Logged in')"]
* on-run-end: "Insert into dbt\_raw.hooks.dbt\_audits (audit\_type) values ('Logged out')"

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

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

  - "target"

  - "dbt\_packages"

## pre/post hooks

if we want to define hooks for all the models then we can define it in the models in the dbt\_project. So in the models if we have different folders for models we can specify it according to the separate folders and inside the folders.

When we will call the folders to run then these hooks will run instead with every and everywhere run like it does in on-run-end/on-run-start

This project level hooks are used to apply hooks to all the models in the project, whether all the models to 1 hooks or according to the folders.

models:

  my\_new\_project:

    # Applies to all files under models/example/

    example:

      chicagocrime:

        chicagocrimedata:

          +pre-hook:

            ["truncate table dbt\_raw.hooks.dbt\_audits", "Insert into dbt\_raw.hooks.dbt\_audits (audit\_type) values ('chicagocrime model pre-hook')"]

          materialized: table

          schema: chic\_crime

          tags:

            - crime

          tests:

      raw\_zone:

        schema: raw\_zone

        tags:

          - healthcare

          - health

        post-hook:

          - "INSERT INTO dbt\_raw.hooks.dbt\_audits (audit\_type) VALUES ('healthcare folder post-hook')"

## config hooks for specific model

if we want to define hooks in a specific and a separate model then it can be defined in the config block of the model.

We can use both pre and post hooks here.

When the particular model is run then that time these hooks will run not like the one defined in dbt\_project in models that calling folder will trigger it or calling any model in that models will trigger it

It will only be triggered when that particular model is being run

{{ config(tags=["crime"], schema = "chic",

 pre\_hook = ["truncate table dbt\_raw.hooks.dbt\_audits", "Insert into dbt\_raw.hooks.dbt\_audits (audit\_type) values ('models sep conf log')"]) }}

with chicagocrime as (select \* from {{ source('src\_chicagocrime', 'chicagocrimedata') }})

select \*

from chicagocrime