Aliases

In dbt (data build tool), aliases and custom aliases are used to provide alternative names or labels for database tables, columns, or other database objects. They allow you to define more meaningful or user-friendly names for these objects within your dbt project. Here's a detailed explanation of aliases and custom aliases in dbt:

1. Aliases: Aliases in dbt provide a way to assign a different name to an existing database object, typically a table or a column. They can be useful when the original object names are not intuitive or when you want to use a different naming convention in your dbt project. Aliases are defined in the **dbt\_project.yml** file under the **models** section. Here's an example:

models: - name: my\_table

alias: my\_alias

In this example, the **my\_table** in your source database will be referred to as **my\_alias** within your dbt project. You can use this alias in your SQL code when referring to the table.

When dbt creates a relation ([table](https://docs.getdbt.com/terms/table)/[view](https://docs.getdbt.com/terms/view)) in a database, it creates it as: **{{ database }}.{{ schema }}.{{ identifier }},** e.g. analytics.finance.payments

The standard behavior of dbt is:

* *If a custom alias is not specified,* ***the identifier of the relation is the resource name (i.e. the filename).***
* *If a custom alias is specified, the identifier of the relation is the****{{ alias }}****value.*

Overview[​](https://docs.getdbt.com/docs/build/custom-aliases#overview)

When dbt runs a model, it will generally create a relation (either a table or a view) in the database. By default, dbt uses the filename of the model as the identifier for this relation in the database. This identifier can optionally be overridden using the alias model configuration.

Why alias model names?[​](https://docs.getdbt.com/docs/build/custom-aliases#why-alias-model-names)

The names of schemas and tables are effectively the "user interface" of your [data warehouse](https://docs.getdbt.com/terms/data-warehouse). Well-named schemas and tables can help provide clarity and direction for consumers of this data. In combination with [custom schemas](https://docs.getdbt.com/docs/build/custom-schemas), model aliasing is a powerful mechanism for designing your warehouse.

Usage[​](https://docs.getdbt.com/docs/build/custom-aliases#usage)

The alias config can be used to change the name of a model's identifier in the database. The following [table](https://docs.getdbt.com/terms/table) shows examples of database identifiers for models both with, and without, a supplied alias.

| **Model** | **Config** | **Database Identifier** |
| --- | --- | --- |
| ga\_sessions.sql | <None> | "analytics"."ga\_sessions" |
| ga\_sessions.sql | {{ config(alias='sessions') }} | "analytics"."sessions" |

To configure an alias for a model, supply a value for the model's alias configuration parameter. For example:

**models/google\_analytics/ga\_sessions.sql**  
*-- This model will be created in the database with the identifier `sessions`*  
*-- Note that in this example, `alias` is used along with a custom schema*  
{{ **config(alias='sessions'**, schema='google\_analytics') }}  
  
select \* from ...

* ***When referencing the ga\_sessions model above from a different model, use the ref() function with the model's filename as usual.***

For example:

**models/combined\_sessions.sql**

*-- Use the model's filename in ref's, regardless of any aliasing configs*  
select \* from **{{ ref('ga\_sessions') }}**  
union all  
select \* from **{{ ref('snowplow\_sessions') }}**

Real example:

1. first of ALL, assign an alias to the model or table you want

{{ config(**alias = "crime\_report",**

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')"]) }}

1. now run the dbt command to create a table or run a model to store it in the warehouse

dbt run –s chicagocrimedata – we will have to run the model with its original name and not aliased name

1. now let us check where it has store/created table with which name

created sql table model dbt\_raw.star\_schema\_chic.**crime\_report**

hence model has to be run with its original name and it will store the table with alias name assigned to it.

imp

When alias name is defined in the configuration it will store that table alias name like in order of

database.schema.alias but when it is called it will be called or referenced with its original name

original : dbt\_raw.star\_schema\_chic.**chicagocrimedata**

alias name: dbt\_raw.star\_schema\_chic.**crime\_report**

when REFERENCED: select \* from **dbt\_raw.star\_schema\_chic.chicagocrimedata**

will be used.

Problem Statement

**Caveats**[**​**](https://docs.getdbt.com/docs/build/custom-aliases#caveats)

**Ambiguous database identifiers**[**​**](https://docs.getdbt.com/docs/build/custom-aliases#ambiguous-database-identifiers)

Using aliases, it's possible to accidentally create models with ambiguous identifiers. Given the following two models, dbt would attempt to create two [views](https://docs.getdbt.com/terms/view) with *exactly* the same names in the database (ie. **sessions**):

***-- models/snowplow\_sessions.sql*  
  
{{ config(alias='sessions') }}**  
  
select \* from ...

***-- models/sessions.sql***  
  
select \* from ...

*Whichever one of these models runs second would "win", and generally, the output of dbt would not be what you would expect. To avoid this failure mode, dbt will check if your model names and aliases are ambiguous in nature. If they are, you will be presented with an error message like this:*

**$ dbt compile**

*Encountered an error:  
Compilation Error  
 dbt found two resources with the database representation "analytics.sessions".  
 dbt cannot create two resources with identical database representations. To fix this,  
 change the "schema" or "alias" configuration of one of these resources:  
 - model.my\_project.snowplow\_sessions (models/snowplow\_sessions.sql)  
 - model.my\_project.sessions (models/sessions.sql)*

***If these models should indeed have the same database identifier, you can work around this error by configuring a***[***custom schema***](https://docs.getdbt.com/docs/build/custom-schemas)***for one of the models.***

generate\_alias\_name

The alias generated for a model is controlled by a macro called generate\_alias\_name. This macro can be overridden in a dbt project to change how dbt aliases models. This macro works similarly to the [generate\_schema\_name](https://docs.getdbt.com/docs/build/custom-schemas" \l "advanced-custom-schema-configuration) macro.

To override dbt's alias name generation, create a macro named generate\_alias\_name in your own dbt project. The generate\_alias\_name macro accepts two arguments:

1. The custom alias supplied in the model config
2. The node that a custom alias is being generated for

The default implementation of generate\_alias\_name simply uses the supplied alias config (if present) as the model alias, otherwise falling back to the model name. This implementation looks like this:

**{% macro generate\_alias\_name(custom\_alias, node) %}**

**{% if custom\_alias %}**

**{{ custom\_alias }}**

**{% else %}**

**{{ node.name }}**

**{% endif %}**

**{% endmacro %}**