

# Creating and generating dbt documentation

INTRODUCTION TO DBT



**Mike Metzger**  
Data Engineer

# Why document?

- Sharing data details with other consumers
- Centralize sources of documentation
- Providing details for updates / changes / etc
- Creating examples, suggestions for use, SLA details



# Creating documentation in dbt

- Can provide documentation with model definitions
- Can add documentation about columns within models
- Automatically show data lineage / DAG
- Document any test / validations
- View generated warehouse information
  - Column data types
  - Data sizes

```
version: 2
```

```
models:
```

- name: taxi\_rides\_raw  
description: Yellow Taxi raw data  
access: public
- name: avg\_fare\_per\_day  
description: Average ride per day  
access: public

# Generating documentation in dbt

- `dbt docs`
  - `dbt docs -h`
  - `dbt docs generate`
- Creates the documentation website based on project
- Should be run after `dbt run`

# Accessing documentation

- Web browser
- `dbt docs serve`
  - Should only be used locally / for development
- Copy content to other hosting service
  - dbt cloud
  - Amazon S3
  - Nginx / Apache / etc

The screenshot displays the dbt documentation interface. At the top left is the dbt logo. To its right is a search bar labeled 'Search for models...'. Below the logo, there are two tabs: 'Project' (selected) and 'Database'. Under the 'Project' tab, a tree view shows the project structure: 'nyc\_yellow\_taxi' (folder), 'models' (folder), 'taxi\_rides' (folder), 'avg\_fare\_per\_day' (file, highlighted), and 'taxi\_rides\_raw' (file). The main content area on the right is titled 'avg\_fare\_per\_day view'. It has several sub-sections: 'Details' (with links for Details, Description, Columns, Depends On, and Code), 'Details' (a table with columns: TAGS, OWNER, TYPE, PACKAGE, LANGUAGE; rows: untagged, view, nyc\_yellow\_taxi, sql), 'Description' (text: 'The average ride amount spent per day'), 'Columns' (a table with columns: COLUMN, TYPE, DESCRIPTION, TESTS; rows: day, BIGINT, avg\_amount, DOUBLE), and 'Depends On'.

dbt

Search for models...

Overview

Project Database

Projects

- nyc\_yellow\_taxi
- models
  - taxi\_rides
    - avg\_fare\_per\_day
    - taxi\_rides\_raw

avg\_fare\_per\_day view

Details Description Columns Depends On Code

Details

TAGS	OWNER	TYPE	PACKAGE	LANGUAGE
untagged		view	nyc_yellow_taxi	sql

Description

The average ride amount spent per day

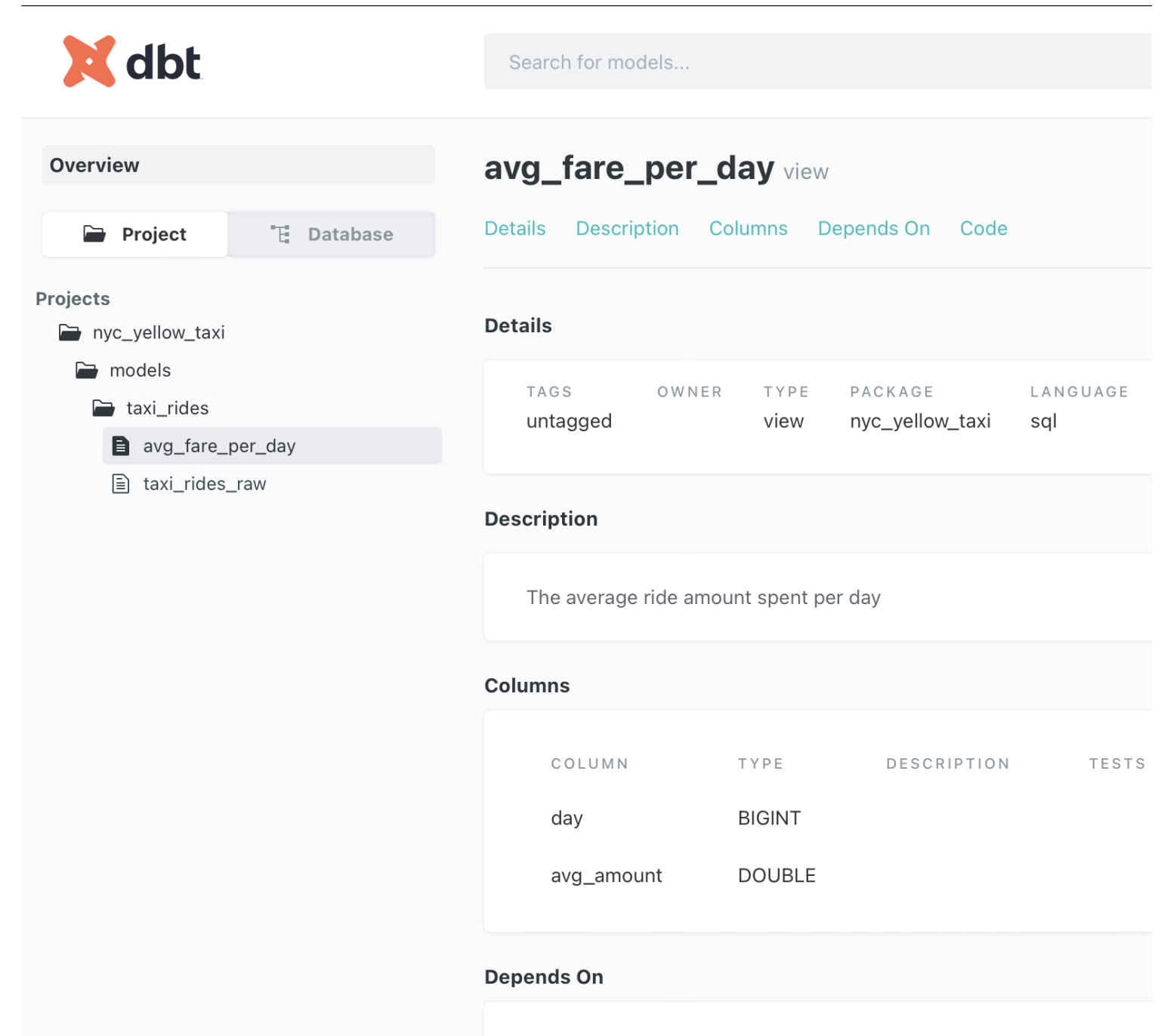
Columns

COLUMN	TYPE	DESCRIPTION	TESTS
day	BIGINT		
avg_amount	DOUBLE		

Depends On

# Documentation example

- View
  - Models
  - Description information
  - Column details
  - Lineage graphs



The screenshot shows the dbt documentation interface. At the top left is the dbt logo. To its right is a search bar labeled "Search for models...". Below the logo is a navigation sidebar with an "Overview" tab and two sub-tabs: "Project" and "Database". Under "Project", there is a tree view showing the project structure: "nyc\_yellow\_taxi" (folder), "models" (folder), "taxi\_rides" (folder), "avg\_fare\_per\_day" (file, highlighted), and "taxi\_rides\_raw" (file). The main content area is titled "avg\_fare\_per\_day view". It has a sub-navigation bar with tabs: "Details", "Description", "Columns", "Depends On", and "Code". The "Details" tab is active, showing a table with the following data:

TAGS	OWNER	TYPE	PACKAGE	LANGUAGE
untagged		view	nyc_yellow_taxi	sql

Below the table is the "Description" section, which contains the text: "The average ride amount spent per day". Below that is the "Columns" section, which contains a table with the following data:

COLUMN	TYPE	DESCRIPTION	TESTS
day	BIGINT		
avg_amount	DOUBLE		

Below the columns table is the "Depends On" section, which is currently empty.

# Let's practice!

INTRODUCTION TO DBT

# Jinja templates

INTRODUCTION TO DBT



**Mike Metzger**  
Data Engineer



# What is a template?

- A defined format allowing for substitution of information
- Often a text file
- Simplifies access and re-use

# What is Jinja?

- Simple text based templating language
- Used in many tools beyond just dbt
- `{{ ... }}` represents a template substitution
- dbt has many Jinja functions available for use in projects
- Allows for more dynamic usage of dbt
- Can use for loops to simplify code

# dbt Jinja functions

- Some of the many Jinja functions available
  - `ref` : Access models by name
  - `config` : Access configuration settings
  - `docs` : Access documentation information
  - Calculations, Loops, etc

# Jinja example

- Without Jinja

```
SELECT
COALESCE(start_date, '2025-01-01') as start_date,
COALESCE(update_date, '2025-01-01') as update_date,
COALESCE(end_date, '2025-01-01') as end_date
FROM Events
```

- With Jinja

```
SELECT
{% for column in ['start_date', 'update_date', 'end_date'] %}
COALESCE({{column}}, '2025-01-01') as {{column}}
{% endfor %}
FROM Events
```

# Let's practice!

INTRODUCTION TO DBT

# Hierarchical models in dbt

INTRODUCTION TO DBT



**Mike Metzger**  
Data Engineer

# What is a hierarchy in dbt?

- Represents the dependencies within a dbt project
- Also known as a directed acyclic graph (DAG), or lineage graph
  - Note this being specific to dbt, rather than general DAG
- Allows models to be built / updated according to dependencies



# Hierarchy details

- `avg_fare_per_day` and `total_creditcard_riders_per_day` depend on `taxi_rides_raw`
- dbt builds `taxi_rides_raw` first
- dbt creates entries alphabetically without a lineage graph
  - `avg_fare_per_day` would fail without `taxi_rides_raw`





# How are hierarchies defined?

- Built using the Jinja template language in the model file(s)
- Most often using the `ref` function
- Replace table name with `{{ ref('model_name') }}` in SQL
- (Re-)Materialize table with `dbt run`
- `ref` templates are substituted with actual table names

# Hierarchy example

**SELECT**

first\_name, last\_name

**FROM** taxi\_rides\_raw

**SELECT**

first\_name, last\_name

**FROM** {{ ref('taxi\_rides\_raw') }}

# Let's practice!

INTRODUCTION TO DBT