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

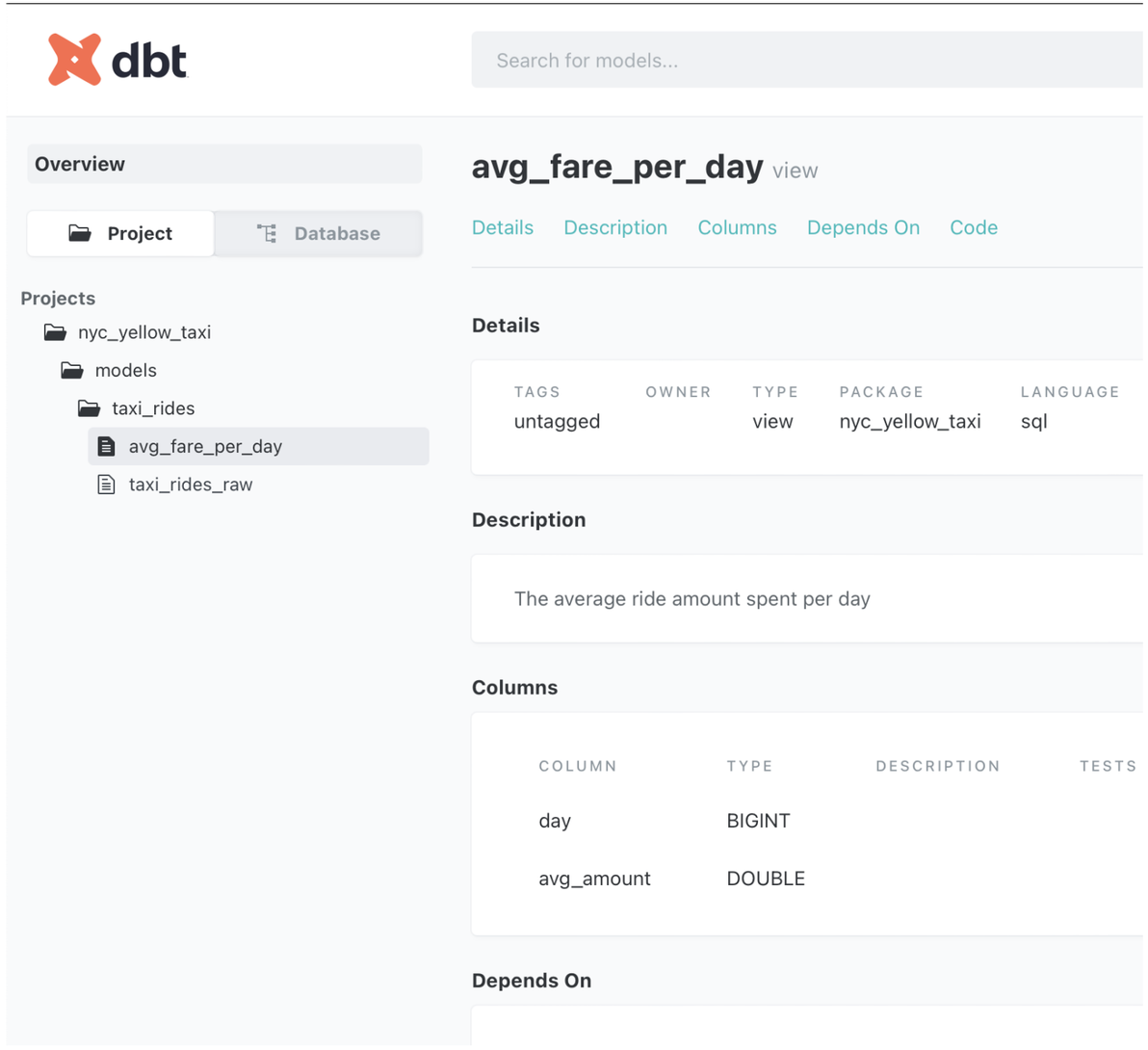
```
version: 2

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`

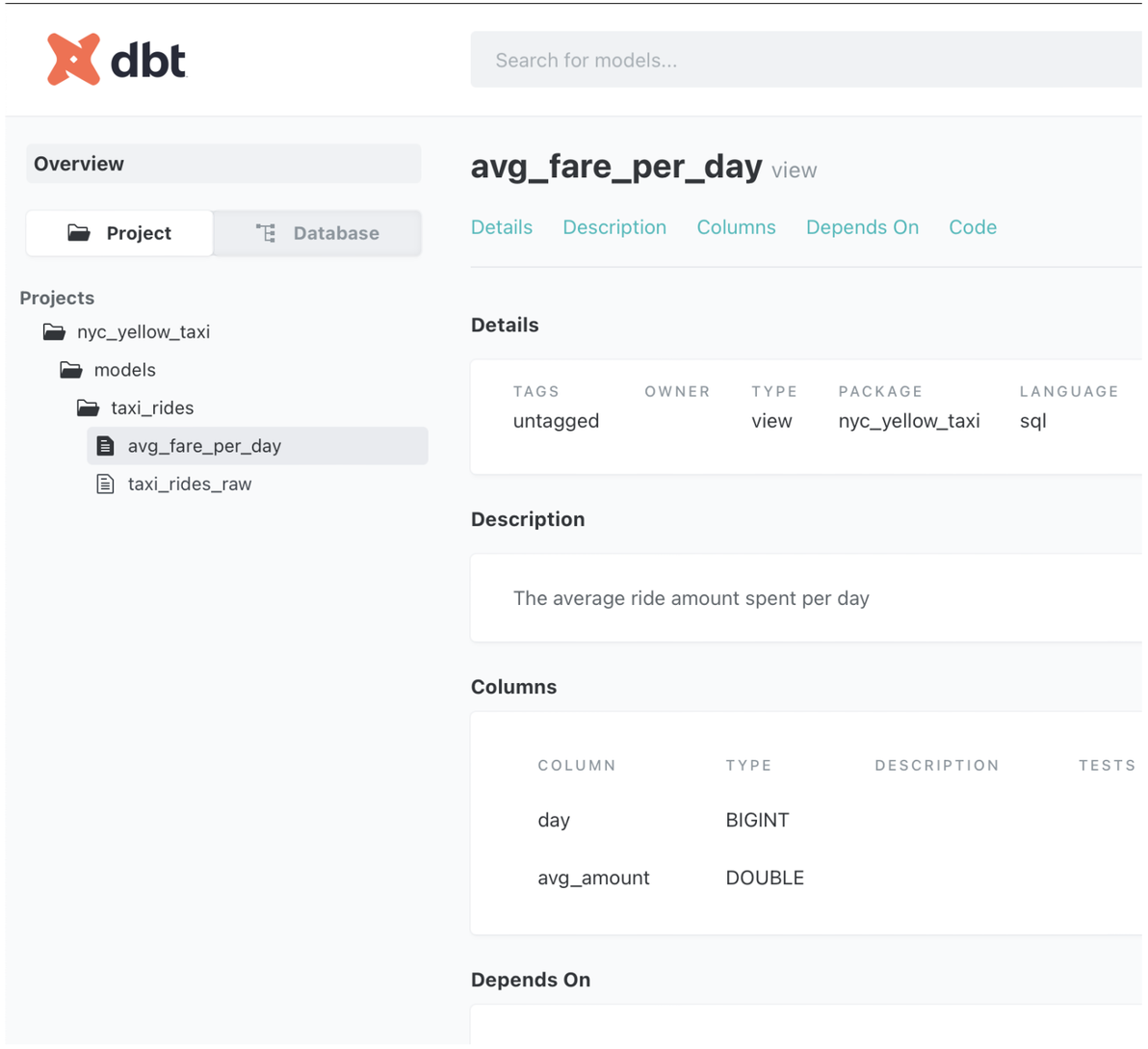
Accessing documentation

- 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



Documentation example

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



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

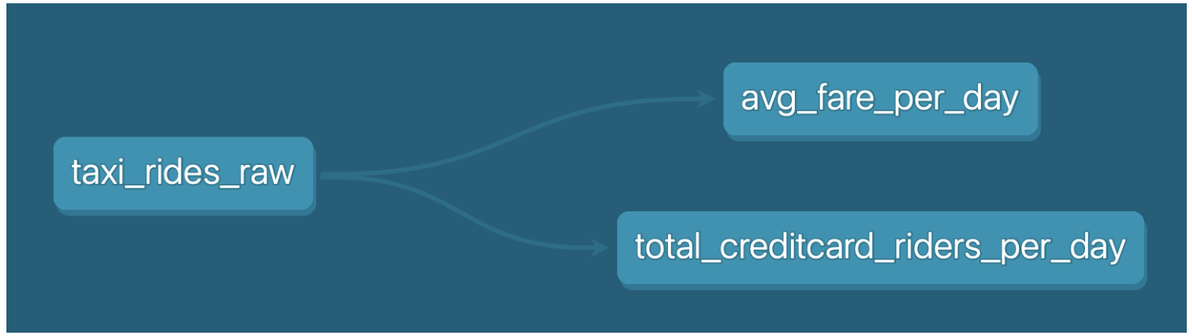
Hierarchical models in dbt

INTRODUCTION TO DBT

Mike Metzger
Data Engineer

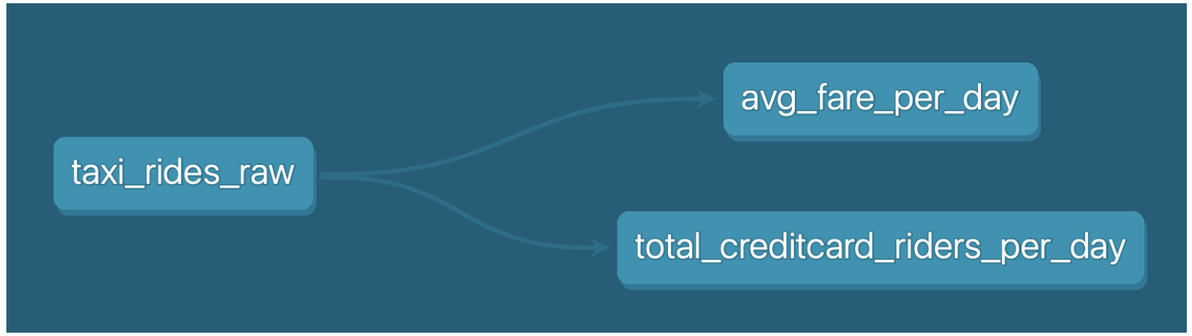
What is a hierarchy in dbt?

- 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