Working with a first project

INTRODUCTION TO DBT

Mike Metzger
Data Engineer

datacamp

Workflow for dbt

1. Create project (`dbt init`)
2. Define configuration (`profiles.yml`)
3. Create / use models / templates
4. Instantiate models (`dbt run`)
5. Verify / Test / Troubleshoot
6. Repeat as needed

dbt run

- Run whenever there model changes
- Or when the data process needs to be materialized
- Output provides many details on the success or failure of the various steps
- Materialized = Transformations into tables / views

```
repl:~$ dbt run
```

```
04:52:11  Running with dbt=1.8.4
...
04:52:13  1 of 1 START sql view model main.sales_data ....... [RUN]
04:52:13  1 of 1 OK created sql view model main.sales_data . [OK in 0.12s]
...
04:52:13  Completed successfully
```

Table vs View

Tables:

- Objects within a database / warehouse that hold data
- Take up space within the database
- Content only updated when changed

- Can be created by dbt

Views:

- Queryable like a table; hold no information
- Are usually defined as a select query against another table or tables
- Content generated with each query

- Can be created by dbt

Let's practice!

INTRODUCTION TO DBT

What is a dbt model?

INTRODUCTION TO DBT

Mike Metzger
Data Engineer

What is a data model?

- Conceptual, with different definitions depending on context
- Represents the logical meaning of data
- How the data and its components relate
- Helps users collaborate

What is a data model?

- Conceptual, with different definitions depending on context
- Represents the logical meaning of data
- How the data and its components relate
- Helps users collaborate

Species	# of legs	Venomous
Cheetah	4	No
Duck	2	No
Platypus	4	Yes
Rattlesnake	0	Yes

What is a model in dbt?

- Represents the various transformations
- Typically written in SQL
  - Newer versions can use Python
- Usually a `SELECT` query
- Each model represented by a text file with .sql extension

Simple dbt model

1. Create a directory in the `models` directory
2. Create a `.sql` file in above directory
3. Add the SQL statement to the newly created file
4. Run `dbt run` to materialize the model

```
bash> mkdir models/order
bash> touch models/order/customer_orders.sql
```

```
select first_name,
       last_name,
       shipping_address,
       item_quantity
from source_table
```

```
bash> dbt run
```

Reading from Parquet

- Parquet?
  - Columnar binary file format
  - DuckDB can read Parquet files directly
  - read_parquet
    - `SELECT * FROM read_parquet('filename.parquet')`
  - Or simply the filename in single quotes
    - `SELECT * FROM 'filename.parquet'`

Let's practice!

INTRODUCTION TO DBT

Updating dbt models

INTRODUCTION TO DBT

Mike Metzger
Data Engineer

Why update?

- Iterative work
- Fixing bugs with queries / models
- Migrating to different sources / destinations

[1] Photo by Caspar Camille Rubin on Unsplash

Update workflow

1. Check out from source control
   - `git clone dbt_project`
2. Find the model in question
3. Update query contents
4. Regenerate with
   - `dbt run` or
   - `dbt run -f` (Force full refresh)
5. Check changes back to source control

YAML files

- Some updates may require changes to YAML / `.yml` files
- Typically would require changes in:
  - `dbt_project.yml`
  - `model_properties.yml`

dbt_project.yml

```yaml
# 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: 'nyc_yellow_taxi'
version: '1.0.0'
config-version: 2

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

# 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"


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

# In this example config, we tell dbt to build all models in the example/
# directory as views. These settings can be overridden in the individual model
# files using the `{{ config(...) }}` macro.
models:
```

dbt_project.yml

- Contains mostly contents related to full project
  - Project name / version
  - Directory locations
- Model materialization settings (global)
- One `dbt_project.yml` file per project

model_properties.yml

- Contain settings that reference models
  - Description
  - Documentation details
  - Much more
- Can actually be named anything (with `.yml`) in models/ subdirectory
- Can have as many files as needed

models > ! model_properties.yml

```yaml
version: 2

models:
  - name: taxi_rides_raw
    description: Initial import of the NYC Yellow Taxi trip data from Parquet source
    access: public
  - name: avg_fare_per_day
    description: The average ride amount spent per day
    access: public
```

Let's practice!

INTRODUCTION TO DBT