

# 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

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
  - Used by many tools to efficiently store data
    - Apache Spark
    - Apache Arrow
    - DuckDB
- 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



<sup>1</sup> 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 / `.yaml` files
- Typically would require changes in:
  - `dbt_project.yaml`
  - `model_properties.yaml`

```
! dbt_project.yaml
1
2 # Name your project! Project names should contain only lowercase characters
3 # and underscores. A good package name should reflect your organization's
4 # name or the intended use of these models
5 name: 'nyc_yellow_taxi'
6 version: '1.0.0'
7 config-version: 2
8
9 # This setting configures which "profile" dbt uses for this project.
10 profile: 'nyc_yellow_taxi'
11
12 # These configurations specify where dbt should look for different types of files.
13 # The 'model-paths' config, for example, states that models in this project can be
14 # found in the "models/" directory. You probably won't need to change these!
15 model-paths: ["models"]
16 analysis-paths: ["analyses"]
17 test-paths: ["tests"]
18 seed-paths: ["seeds"]
19 macro-paths: ["macros"]
20 snapshot-paths: ["snapshots"]
21
22 target-path: "target" # directory which will store compiled SQL files
23 clean-targets:         # directories to be removed by `dbt clean`
24 | - "target"
25 | - "dbt_packages"
26
27
28 # Configuring models
29 # Full documentation: https://docs.getdbt.com/docs/configuring-models
30
31 # In this example config, we tell dbt to build all models in the example/
32 # directory as views. These settings can be overridden in the individual model
33 # files using the `{% config(...) %}` macro.
34 models:
35 |   nyc_yellow_taxi:
```

# 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
1  version: 2
2
3  models:
4    - name: taxi_rides_raw
5      description: Initial import of the NYC Yellow Taxi trip data from Parquet source
6      access: public
7    - name: avg_fare_per_day
8      description: The average ride amount spent per day
9      access: public
```

# 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 between models
- 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



# 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

```
SELECT
    first_name, last_name
FROM taxi_rides_raw
```

```
SELECT
    first_name, last_name
FROM {{ ref('taxi_rides_raw') }}
```



# Jinja templating language

- 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
- Some of the many Jinja functions available
  - `ref`
  - `config`
  - `docs`
  - Many more!

# Let's practice!

INTRODUCTION TO DBT

# Model troubleshooting

INTRODUCTION TO DBT



**Mike Metzger**  
Data Engineer

# Common model issues

- Query errors
  - Syntax errors
  - Logic errors
- Invalid references



<sup>1</sup> Photo byahmad gunnaivi on Unsplash

# Query errors

- Misspellings / syntax issues
- Non-standard SQL
- Common SQL logic issues
  - Not grouping by all non-aggregated columns
  - Incorrect CTEs

```
customers.sql • dbt_project.yml
models > customers.sql
1  with customers as (
2      select
3          id as customer_id,
4          first_name,
5          last_name
6      from jaffle_shop.raw_customers
7  ),
8
9
10 orders as (
11     select
12         id as order_id,
13         user_id as customer_id,
14         order_date,
15         status
16     from jaffle_shop.raw_orders
17 ),
18
19
20 customer_orders as (
21     select
22         customer_id,
23         min(order_date) as first_order_date,
24         max(order_date) as most_recent_order_date,
25         count(order_id) as number_of_orders
26     from orders
27     group by 1
```

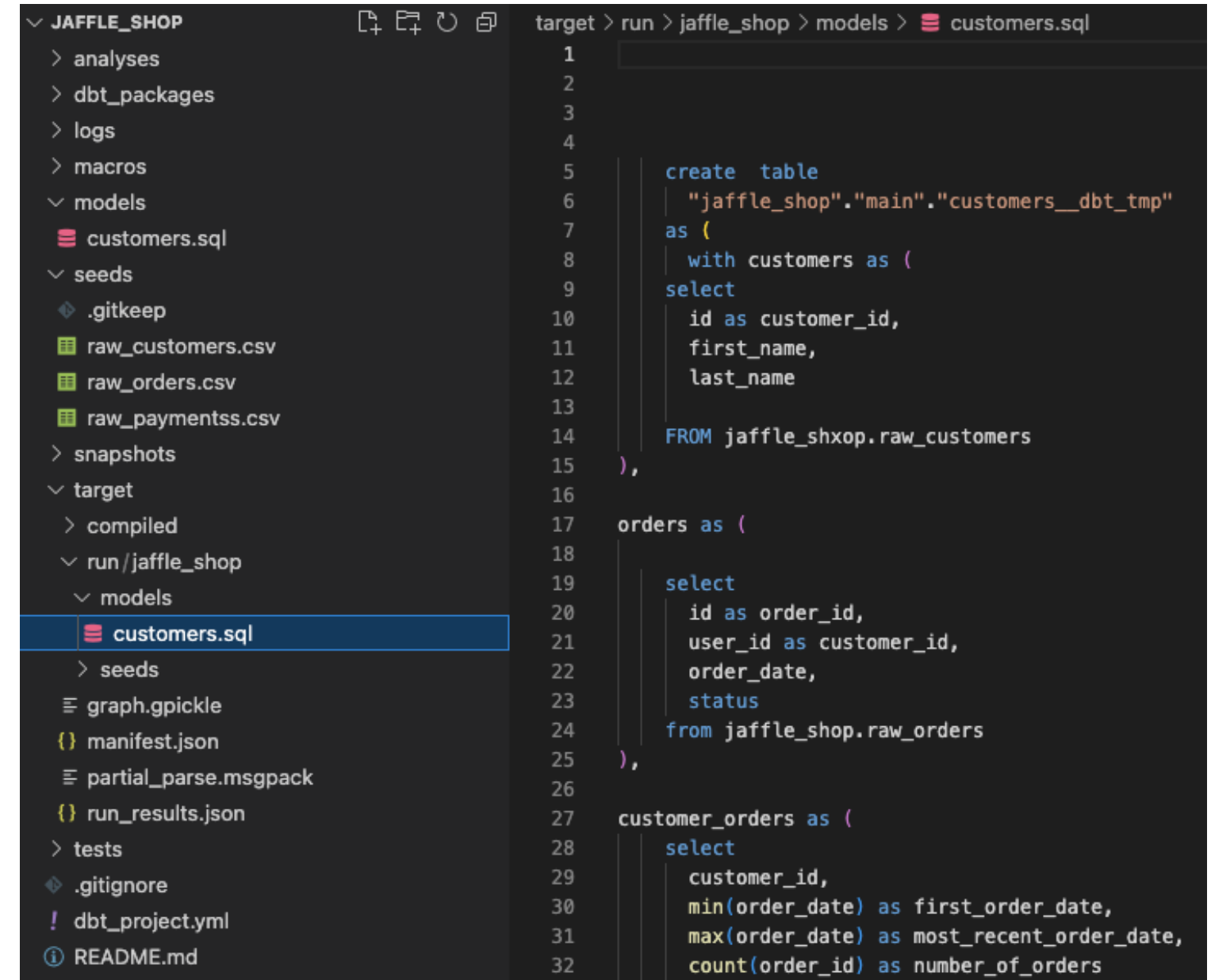
# Invalid references

- Table / view named differently than expected
  - Context name
  - Preceding underscore or periods
- Referencing objects that have yet to be created

```
15:46:21 Running with dbt=1.4.5
15:46:21 Found 1 model, 0 tests, 0 snapshots, 0 analyses, 296 macros, 0 operations, 3 seed files, 0 sources, 0 exposures, 0 metrics
15:46:21
15:46:21 Concurrency: 1 threads (target='dev')
15:46:21
15:46:21 1 of 1 START sql table model main.customers ..... [RUN]
15:46:21 1 of 1 ERROR creating sql table model main.customers ..... [ERROR in 0.03s]
15:46:21
15:46:21 Finished running 1 table model in 0 hours 0 minutes and 0.07 seconds (0.07s).
15:46:21
15:46:21 Completed with 1 error and 0 warnings:
15:46:21
15:46:21 Runtime Error in model customers (models/customers.sql)
15:46:21   Catalog Error: Table with name raw_customers does not exist!
15:46:21   Did you mean "jaffle_shop.raw_customers"?
15:46:21
15:46:21 Done. PASS=0 WARN=0 ERROR=1 SKIP=0 TOTAL=1
```

# Troubleshooting methods

- `dbt run`
- View logs
  - `logs/dbt.log`
  - `run_results.json`
- Viewing generated SQL
  - Running generated SQL manually
- Verify fixes



The screenshot shows a code editor with a file explorer on the left and a SQL editor on the right. The file explorer shows a project named 'JAFFLE\_SHOP' with a tree structure including analyses, dbt\_packages, logs, macros, models, seeds, snapshots, target, and run/jaffle\_shop. The 'models' directory is expanded, showing 'customers.sql' selected. The SQL editor displays the content of 'customers.sql', which is a SQL model that creates a table, defines a CTE for customers, and another CTE for orders, then joins them into a final customer\_orders table.

```
target > run > jaffle_shop > models > customers.sql
1
2
3
4
5 create table
6   "jaffle_shop"."main"."customers__dbt_tmp"
7 as (
8   with customers as (
9     select
10      id as customer_id,
11      first_name,
12      last_name
13    from jaffle_shop.raw_customers
14   ),
15
16   orders as (
17     select
18      id as order_id,
19      user_id as customer_id,
20      order_date,
21      status
22    from jaffle_shop.raw_orders
23   ),
24
25   customer_orders as (
26     select
27      customer_id,
28      min(order_date) as first_order_date,
29      max(order_date) as most_recent_order_date,
30      count(order_id) as number_of_orders
31    from jaffle_shop.orders
32   )
```

# Let's practice!

INTRODUCTION TO DBT