dbt meetup

From data engineer to analytics engineering

Dan Gooden & Rifat Majumder • 9th of October, 2019

Analytics engineering is the data transformation work that happens between loading data into your warehouse and analyzing it.

- dbt

Agenda

- Who we are
- Leaving data engineering behind
- What is an analytics engineer?
- Airtaskers journey
- Our setup, challenges and learnings
- Tip: Cloning prod with Snowflake and dbt

Dan

Data Lead at Airtasker

Initial background in enterprise data warehouse consulting

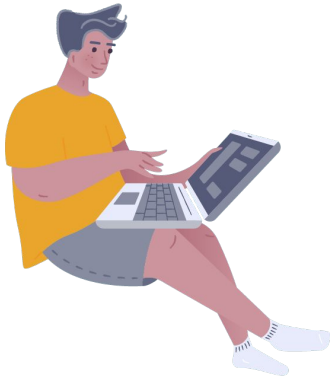
Prior to AT was a Data Platform lead for Domain

Rif

Analytics Engineer at Airtasker

Varied background with last few years in project and product management

Previously at CSIRO/Data61



Leaving data engineering behind

Trends

New tools automating away mundane data engineering tasks

Analysts now have tools they can leverage to fast track data to insights

Trends

These trends help data teams move much **faster** and improve **quality**

Implications

Need to revisit the **composition** of data teams

Data engineering focuses on **data infrastructure**

Analytics engineering focuses on creating **actionable data**

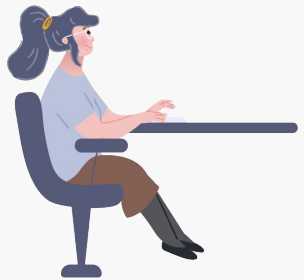
Both roles act as **force multipliers**

Data engineering ⟶ Analytics engineering ⟶ Business Users

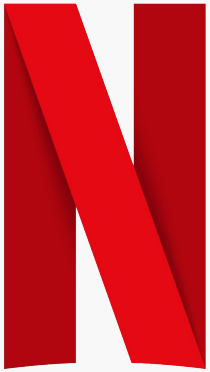
10 x **10 x**

The analytics engineer role

- Strategic partner for business and its various functions
- Technical ownership from the data warehouse through to BI reports and dashboards
- Leverages software engineering practices and tooling approaches



Analytics engineer roles in the wild - Netflix



You are...

- A **strong communicator** who can own and deepen direct relationships with our business partners
- A senior analytics professional with a proven track record of **data wrangling, analysis, reporting & visualization**
- Comfortable with **ambiguity**; able to thrive with minimal oversight and process
- An expert in a **data-oriented programming language** (e.g. SQL, Python, R, Scala, etc.)

Analytics engineer roles in the wild - Pymetrics

Requirements...

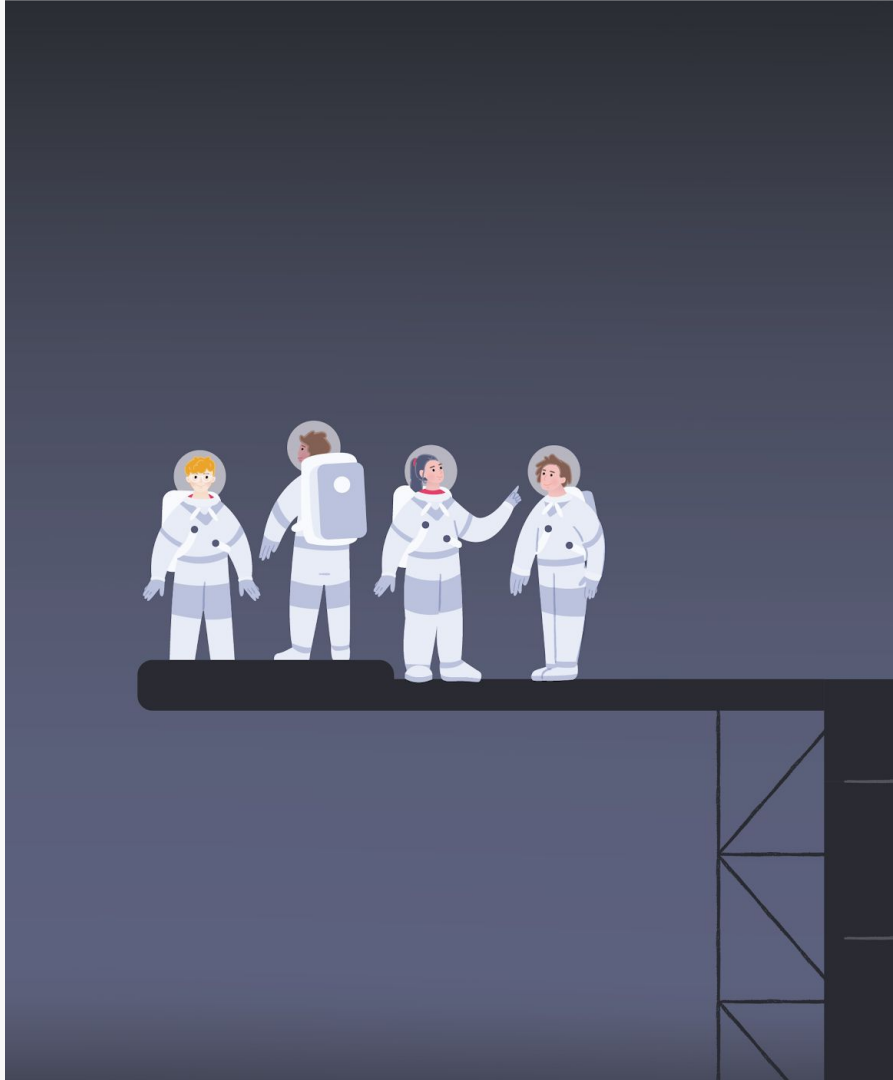
- Experience implementing best-practices for data modelling, especially with regards to **dimensional modelling** for business intelligence. Experience with DBT is a plus.
- Experience with SQL-first, **self-service** visualization tools such as Mode Analytics, Tableau, Looker
- High level of comfort with **SQL** and proficiency in **Python** and its scientific libraries (pandas, scipy etc.)
- Experience with git and **git-based workflows**

Analytics engineer skills

- Software engineering practices (maintainable code, testing, CI/CD)
- SQL patterns
- Data oriented language (e.g. Python, R, Scala)
- Git and bash
- Database fundamentals and performance optimisation
- Data modelling (or dimensional modelling)
- Visual data communication
- Business analysis

Data Tribe @ Airtasker

- 3 x Analytics Engineers
- 2 x Data Scientists
- 1 x Software Engineer
- 1 x Data Analyst
- 1 x Product Manager



Airtasker's (my) Journey

Domain & Luigi

- Custom built ETL
- Processing real time event streams
- Custom load strategies
- Python skills
- Customised orchestration and pipelines

Airtasker & Airflow

- Custom built ETL
- Processing real time event streams
- Custom load strategies
- Python skills
- Customised orchestration and pipelines

Airtasker's (my) Journey

Domain & Luigi

- Custom built ETL
- Processing real time event streams
- Custom load strategies
- Python skills
- Customised orchestration and pipelines

Airtasker & Airflow

- ~~Custom built ETL~~
- Processing real time event streams
- Custom load strategies
- Python skills
- Customised orchestration and pipelines

Airtasker's (my) Journey

Domain & Luigi

- Custom built ETL
- Processing real time event streams
- Custom load strategies
- Python skills
- Customised orchestration and pipelines

Airtasker & Airflow

- ~~Custom built ETL~~
- ~~Processing real time event streams~~
- Custom load strategies
- Python skills
- Customised orchestration and pipelines

Airtasker's (my) Journey

Domain & Luigi

- Custom built ETL
- Processing real time event streams
- Custom load strategies
- Python skills
- Customised orchestration and pipelines

Airtasker & Airflow

- ~~Custom built ETL~~
- ~~Processing real time event streams~~
- ~~Custom load strategies~~
- ~~Python skills~~
- ~~Customised orchestration and pipelines~~

Airtasker's (my) Journey

Domain & Luigi

- Custom built ETL
- Processing real time event streams
- Custom load strategies
- Python skills
- Customised orchestration

Airtasker & ~~Airflow~~ dbt

- ~~Custom built ETL~~
- ~~Processing real time event streams~~
- ~~Custom load strategies~~
- ~~Python skills~~
- ~~Customised orchestration~~

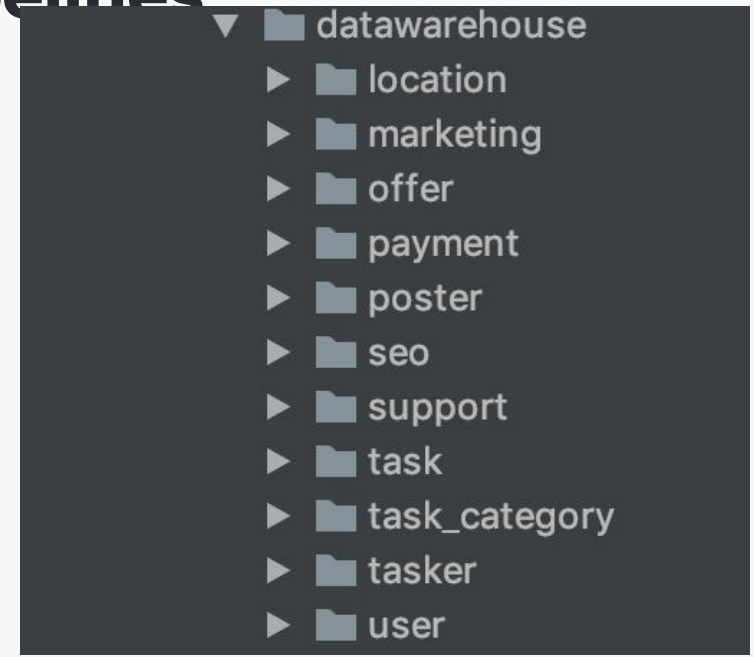
Our Setup

- Run dbt using Docker
- Snowflake is our data warehouse
- Scheduling is done via dbtcloud
- Every Github pull request has a full run and tests performed against it in a CI database which is cloned from prod
- We use macros sparingly as we've found them to decrease readability and clarity of intent in the models
- Most table loads are full refreshes, incremental loads are used sparingly on very large tables (e.g, our event stream)

Learning 1 - Bye bye pipelines

- Independent and separated workflows were a familiar pattern, e.g. a user pipeline, a task pipeline, finance pipeline, hourly pipeline, daily pipeline, etc, scheduled at different times based on source ingestion timeframes

Learning 1 - Bye bye pipelines

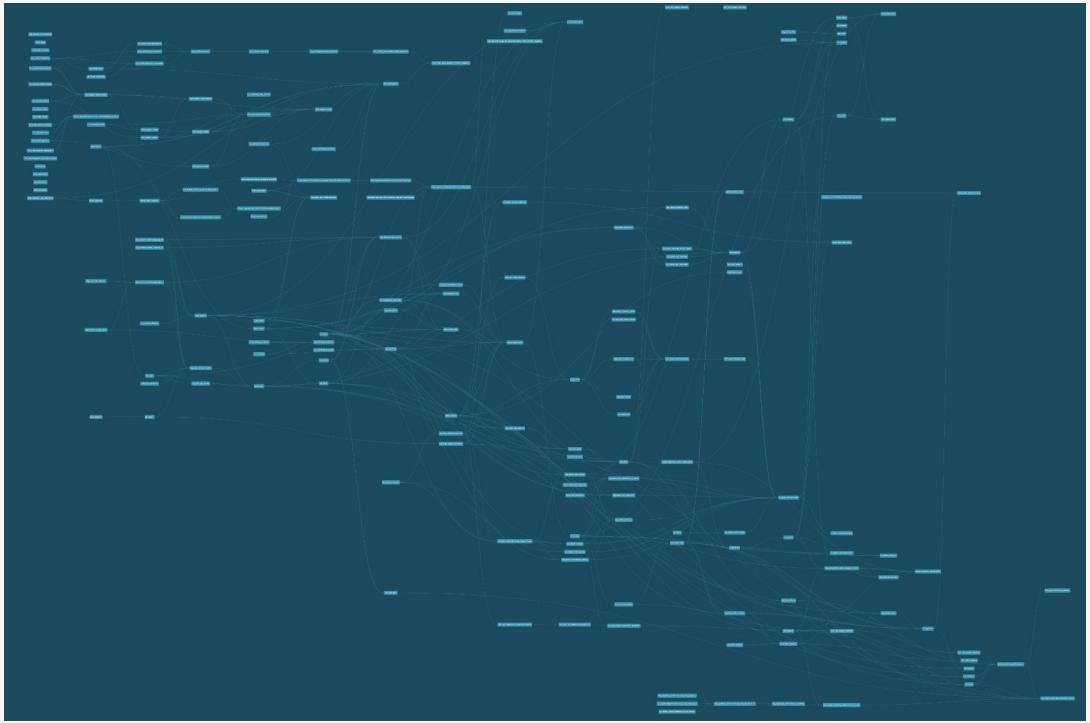


Learning 1 - Bye bye pipelines

- Race conditions created failing tests
- Dbt works best when everything runs together
- We now have a single load that runs every 4 hours

Learning 2 - Thinking in layers

- Our DAG has gotten large and complicated



DATA WAREHOUSE

LAYERS

Datamart layers

Product

The product layer is designed to be consumed by customer facing products. I.e. Buy now removalists.

Feature store

The feature store layer is designed to be consumed by algorithms.

Dimensional modelling

The dimensional modelling layer is our source of truth layer. The tables in this layer should model and describe business processes. They should include segmentation, bucketing, calculated metrics, and any other fields necessary to analyse the process.

In implementation these models should be relatively simple, with many joins. No complex calculations should be needed, aim for single line calculations. Any complex calculations or transformations should be pushed back to the staging layer.

The historical layer captures raw or processed data at a particular point in history. Models and tables here are used to enable and build analytics that can reflect a historical view.

Historical

Staging

Core

This layer is where we build out our core business entities. I.e. we don't have users, we have posters and taskers. No calculations are performed here. We are just getting the data into our shape.

Staging

Source layers

Base

The base layer denormalizes the raw entities, ensuring entities have their missing columns and rows. Otherwise, no logic exists in this layer. E.g. make sure that task has a location_id.

Source

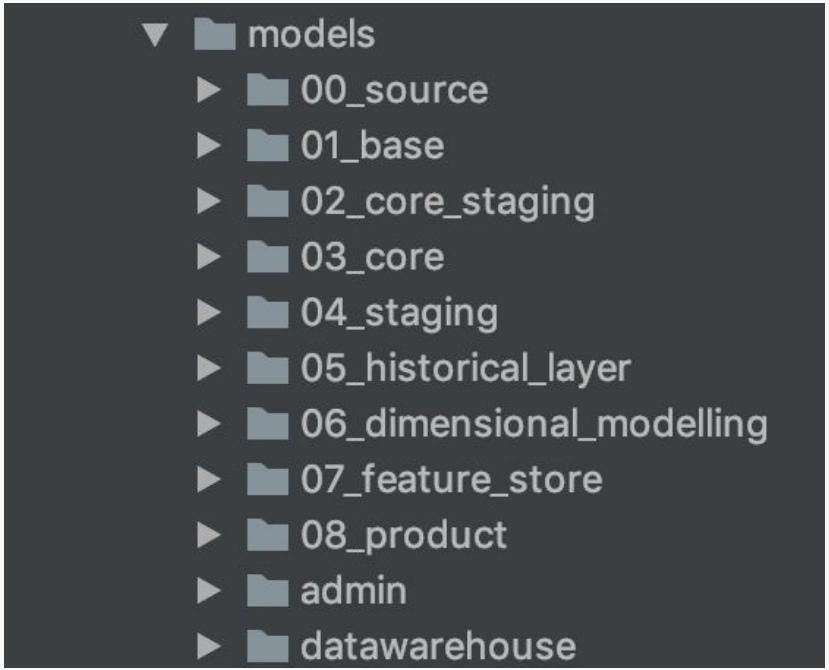
The source layer deals with simple column renaming. No logic exists in this layer. Only include rows from the last full Fivetran sync where applicable.

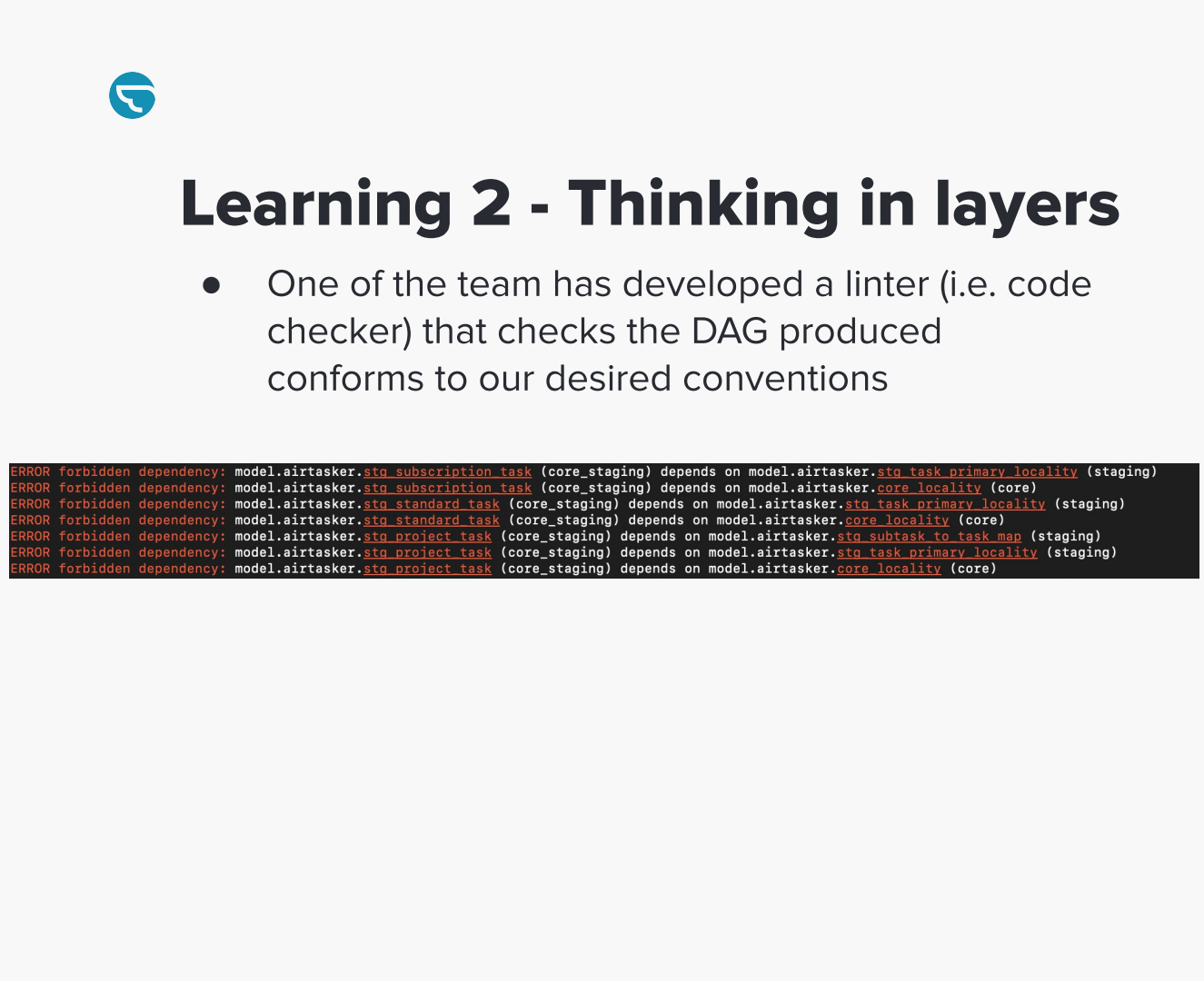
Raw

All raw data from our data sources sits in this layer.

Learning 2 - Thinking in layers

- We restructured our project to align with our thinking



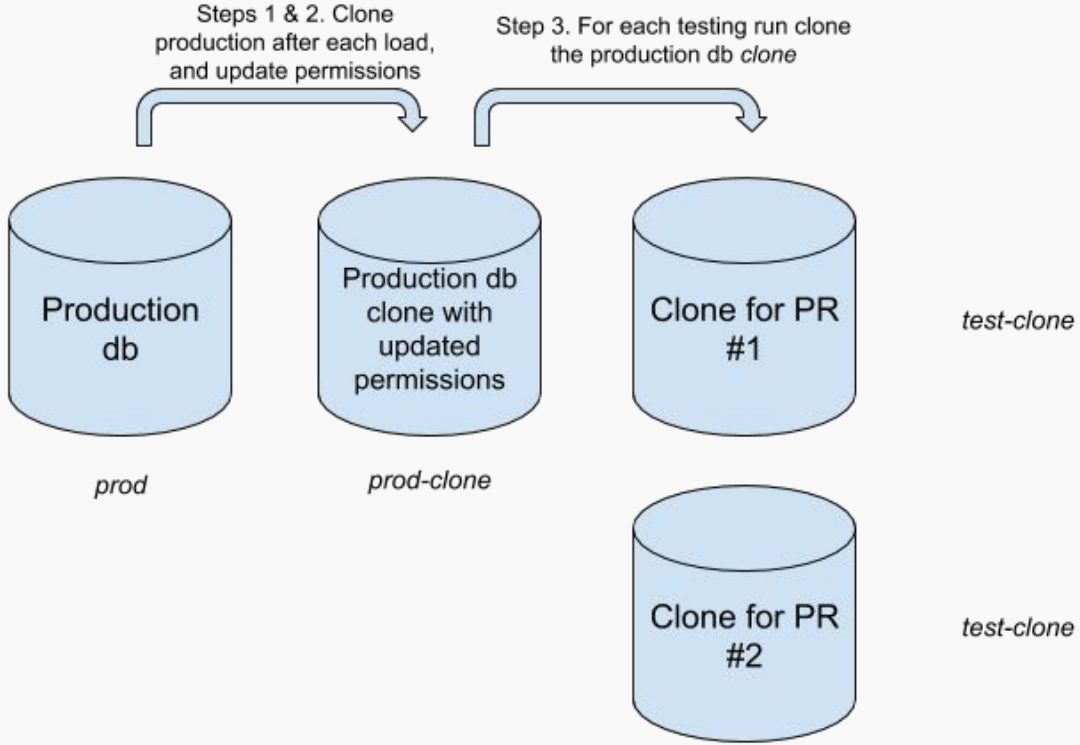


Learning 2 - Thinking in layers

- One of the team has developed a linter (i.e. code checker) that checks the DAG produced conforms to our desired conventions

```
ERROR forbidden dependency: model.airtasker.stg_subscription_task (core_staging) depends on model.airtasker.stg_task_primary_locality (staging)
ERROR forbidden dependency: model.airtasker.stg_subscription_task (core_staging) depends on model.airtasker.core_locality (core)
ERROR forbidden dependency: model.airtasker.stg_standard_task (core_staging) depends on model.airtasker.stg_task_primary_locality (staging)
ERROR forbidden dependency: model.airtasker.stg_standard_task (core_staging) depends on model.airtasker.core_locality (core)
ERROR forbidden dependency: model.airtasker.stg_project_task (core_staging) depends on model.airtasker.stg_subtask_to_task_map (staging)
ERROR forbidden dependency: model.airtasker.stg_project_task (core_staging) depends on model.airtasker.stg_task_primary_locality (staging)
ERROR forbidden dependency: model.airtasker.stg_project_task (core_staging) depends on model.airtasker.core_locality (core)
```

Clone prod with Snowflake & dbt



3 macros

clone_prod_and_update_permissions()
clone_schema_from_prod()
clone_schemas_to_target_db()

Pre-hooks on models for older versions of dbt < 0.14

On the upgrade path, but running into challenges.

How to use pass arguments and variables to operations?

More details

https://medium.com/airtribe

Wrap Up

Dbt is just a tool

dbt collapses the orchestration and development into a single step, meaning you can effectively data model, design and deploy with great efficiency.

But success will *still* be dictated by the quality of your models, and quality of your data

Dbt helps us go faster with more safety

Thank you