**Introduction**

* When a company starts experiencing problems with their data pipelines, the first thing they usually turn to is DBT tests
* DBT tests are good, up to a point
* But often companies stay with DBT tests too long, when instead they should move on to observability

**Testing vs. Observability**

* Explain the difference between testing and observability
* Testing is a human who has knowledge of the data expressing a condition that they know should be true
* observability is unopinionated. It is purely saying, do we know the state of the thing at all times? And do we know how the state is changing? And if the state changes in a way that is concerning, can we know about that?
* A good example is the dashboard on a vehicle. You can still go on a road trip without it, but most people would be pretty uncomfortable not knowing how fast they're going, or if the engine light is on. Something very bad might be happening with the car and you're not going to know about it.
  + It would be silly to test all those things all the time. It makes more sense just to have a constant input of signals about them and when those signals look off, go and take action.
  + if you are purely relying on tests, the only information you've got is the few things you decided to test.

**Can DBT become a full-fledged data observability platform?**

* Talk about what DBT can do, and what it can’t do
* Does unit tests and data freshness checks
* Does not have column lineage or anomaly detection
* Support macros
* OpenLineage
* Plugins and packages for great expectations

**Adding observability to your DBT deployment (Case study)**

https://discourse.getdbt.com/t/how-to-add-observability-to-your-dbt-deployment/3451

**Limitations**

Even after building something like ^^, you have adequate observability into dbt, but ignorant of other parts of the stack. This is where other tools like Bigeye, etc can help, because it looks at the final data at rest after everything has been transformed.