Analyses

**Overview**[**​**](https://docs.getdbt.com/docs/build/analyses#overview)

dbt's notion of models makes it easy for data teams to version control and collaborate on data transformations. Sometimes though, a certain sql statement doesn't quite fit into the mold of a dbt model. These more "analytical" sql files can be versioned inside of your dbt project using the analysis functionality of dbt.

Any .sql files found in the analyses/ directory of a dbt project will be compiled, but not executed. This means that analysts can use dbt functionality like {{ ref(...) }} to select from models in an environment-agnostic way.

In practice, an analysis file might look like this (via the [open source Quickbooks models](https://github.com/dbt-labs/quickbooks)):

analyses/running\_total\_by\_account.sql

*-- analyses/running\_total\_by\_account.sql*  
  
**with journal\_entries as (**  
 **select \*  
 from {{ ref('quickbooks\_adjusted\_journal\_entries') }}  
  
), accounts as (  
 select \*  
 from {{ ref('quickbooks\_accounts\_transformed') }}  
  
)  
select  
 txn\_date,  
 account\_id,  
 adjusted\_amount,  
 description,  
 account\_name,  
 sum(adjusted\_amount) over (partition by account\_id order by id rows unbounded preceding)  
from journal\_entries  
order by account\_id, id**

*To compile this analysis into runnable sql, run:*

**dbt compile**

Then, look for the compiled SQL file **in target/compiled/{project name}/analyses/running\_total\_by\_account.sql**. This sql can then be pasted into a data visualization tool, for instance. Note that no running\_total\_by\_account relation will be materialized in the database as this is an analysis, not a model.