dbt-project

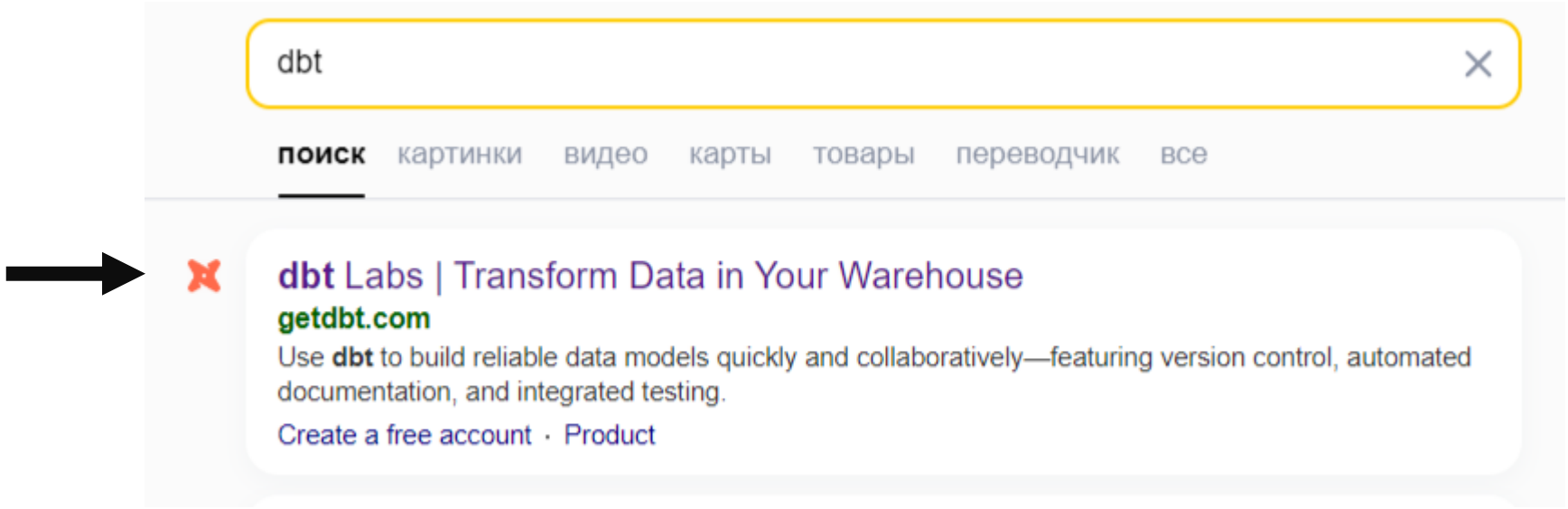
API Youtube/Python + Google Big Query + dbt

Краткое описание проекта

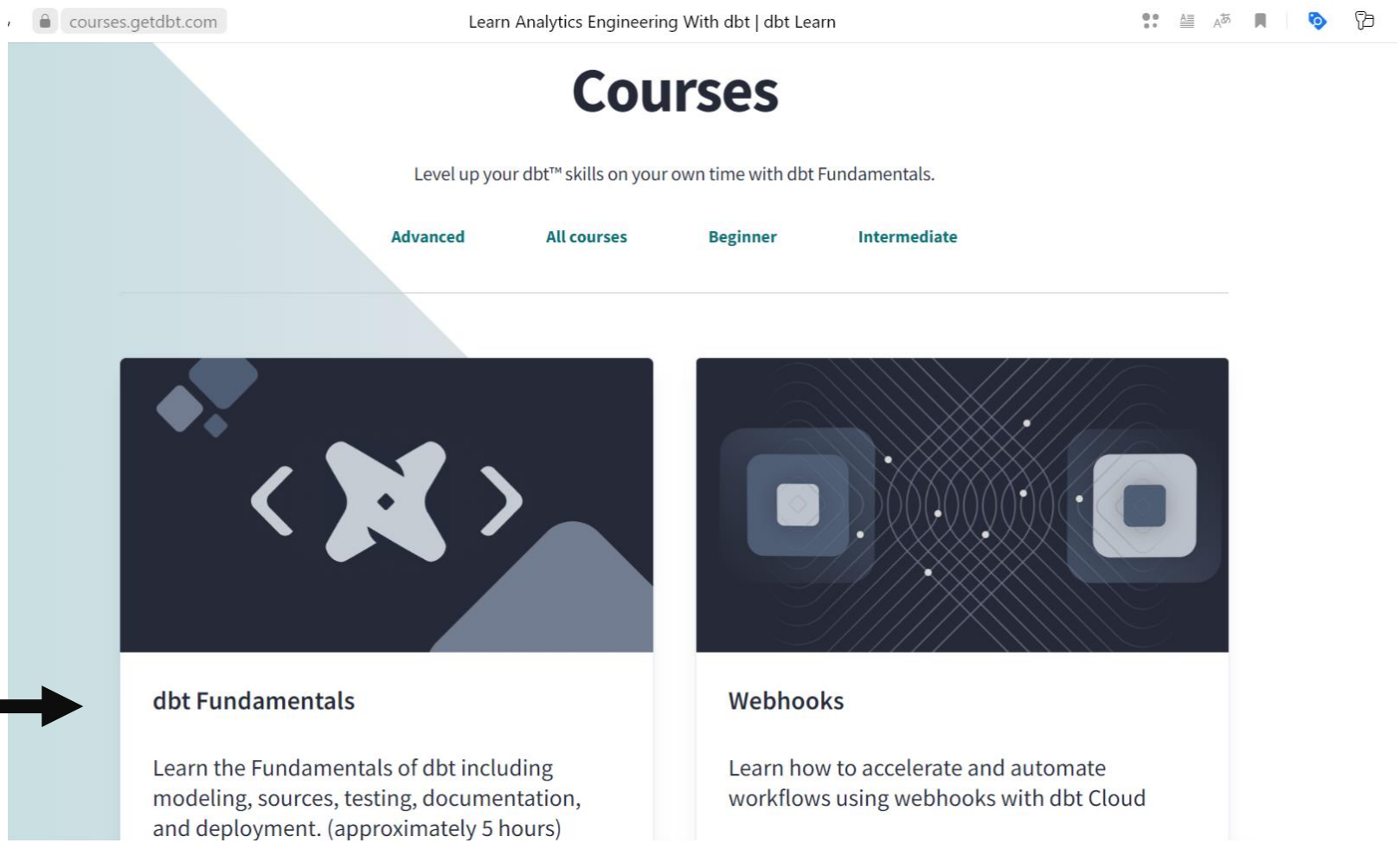
У меня есть серия исследований каналов Youtube. Данный проект – одна из вариаций на эту тему.

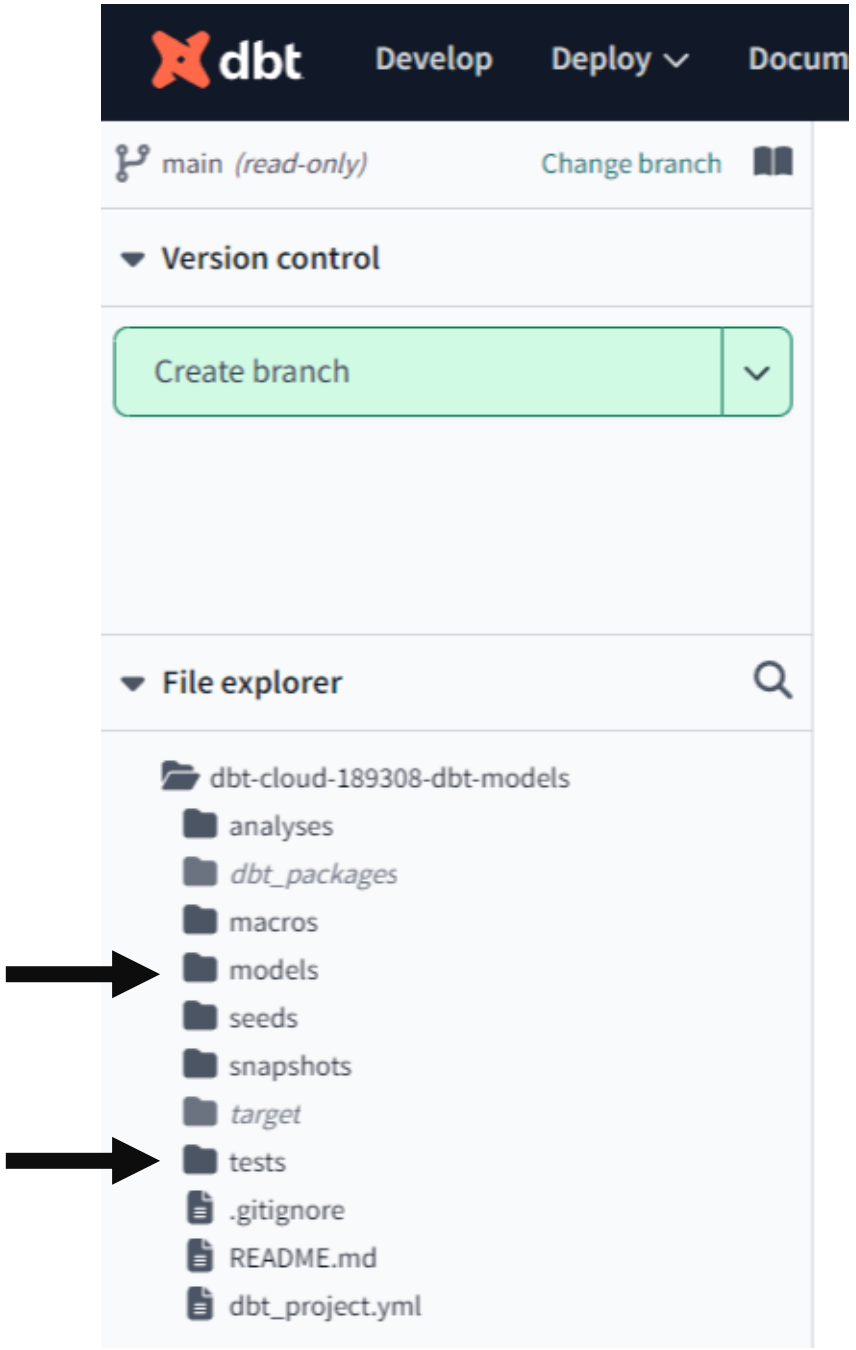
За основу я взяла данные, полученные по ***API Youtube***. Эти несколько датасетов я загрузила в ***Google Big Query***. И они послужили материалом для проекта ***dbt***.

Далее я расскажу и покажу, как устроен мой проект, созданный на **getdbt.com**.



Изучить основы dbt можно на бесплатном курсе dbt Fundamentals:
https://courses.getdbt.com/collections
Этот курс помог мне освоить dbt и создать данный проект.



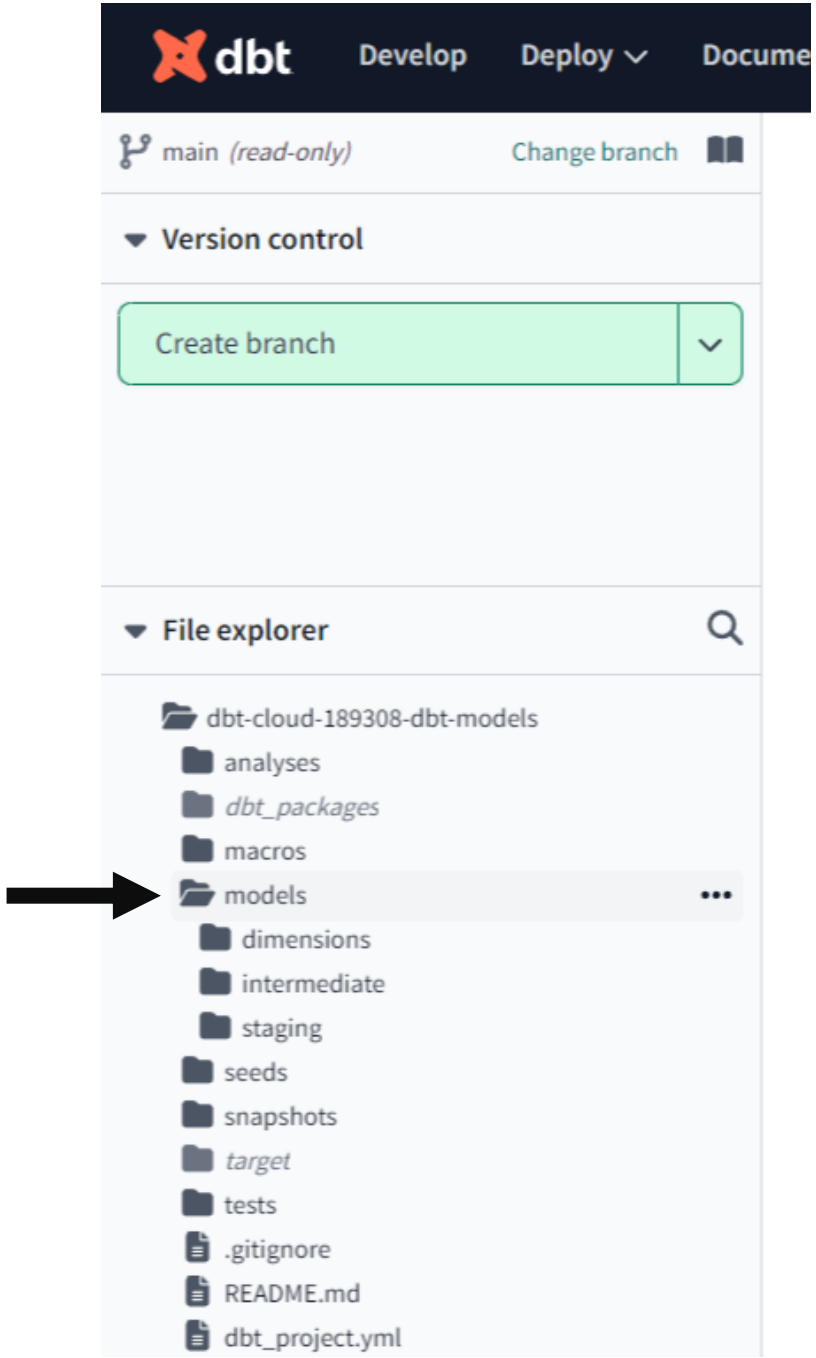


Как выглядит проект:
это несколько папок.

Основные – это models и tests.

models – здесь находятся dbt-модели, разделенные по папкам.
Это – сердцевина всего проекта.

tests – это тесты, они имеют скорее вспомогательную функциональность.
В принципе можно обходиться и вовсе без них (но с ними, всё-таки, интересней).



Рассмотрим подробнее **models**.
Модели внутри я распределила по папкам
для большей наглядности и структурности.

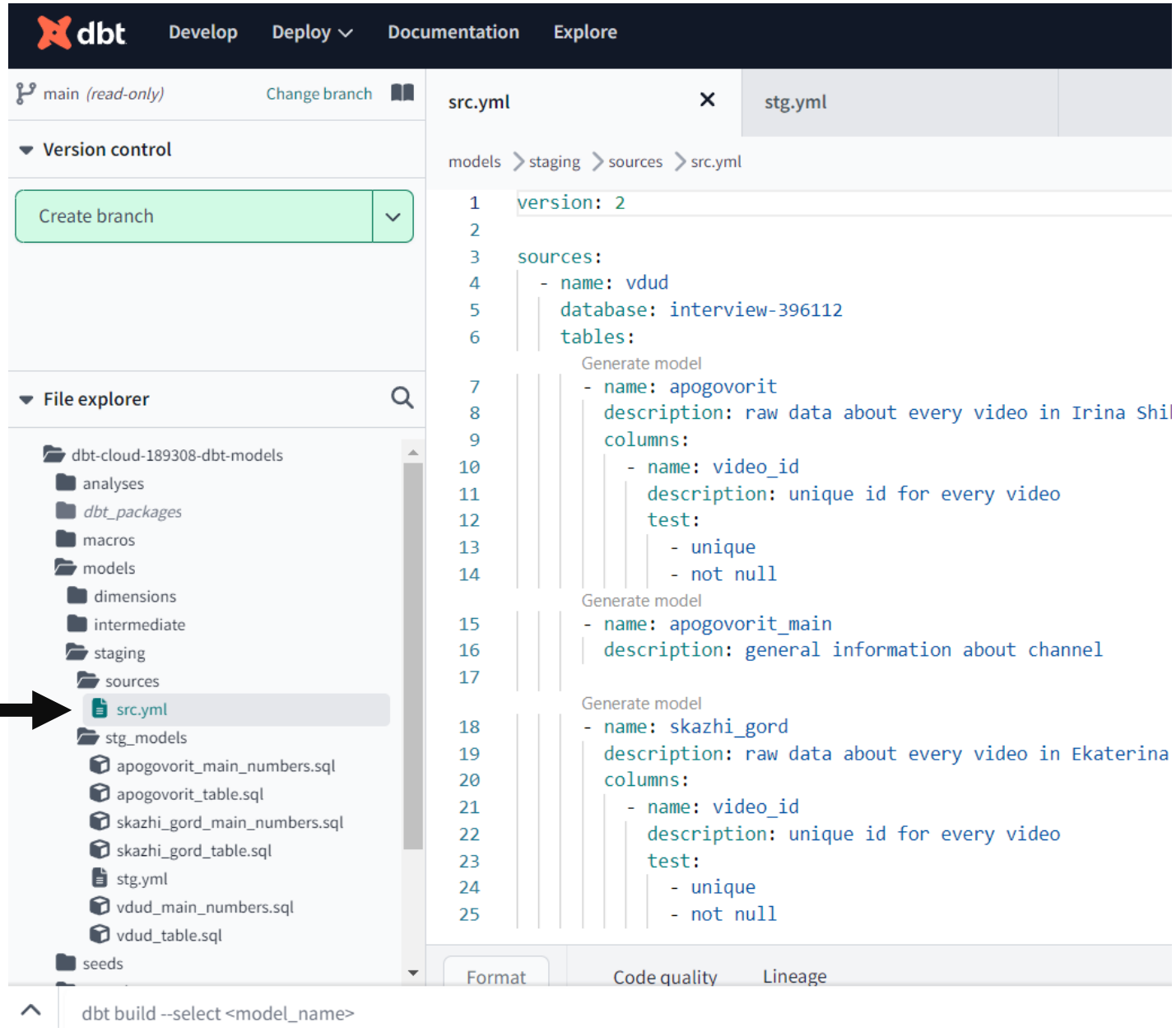
В моём проекте внутри models 3 папки:

- **dimensions**: итоговый слой

- **intermediate**: слой для преобразований и расчётов

- **staging**: слой для сырых данных

Далее рассмотрим проект по пути следования данных:
от «сырого» слоя до документации.



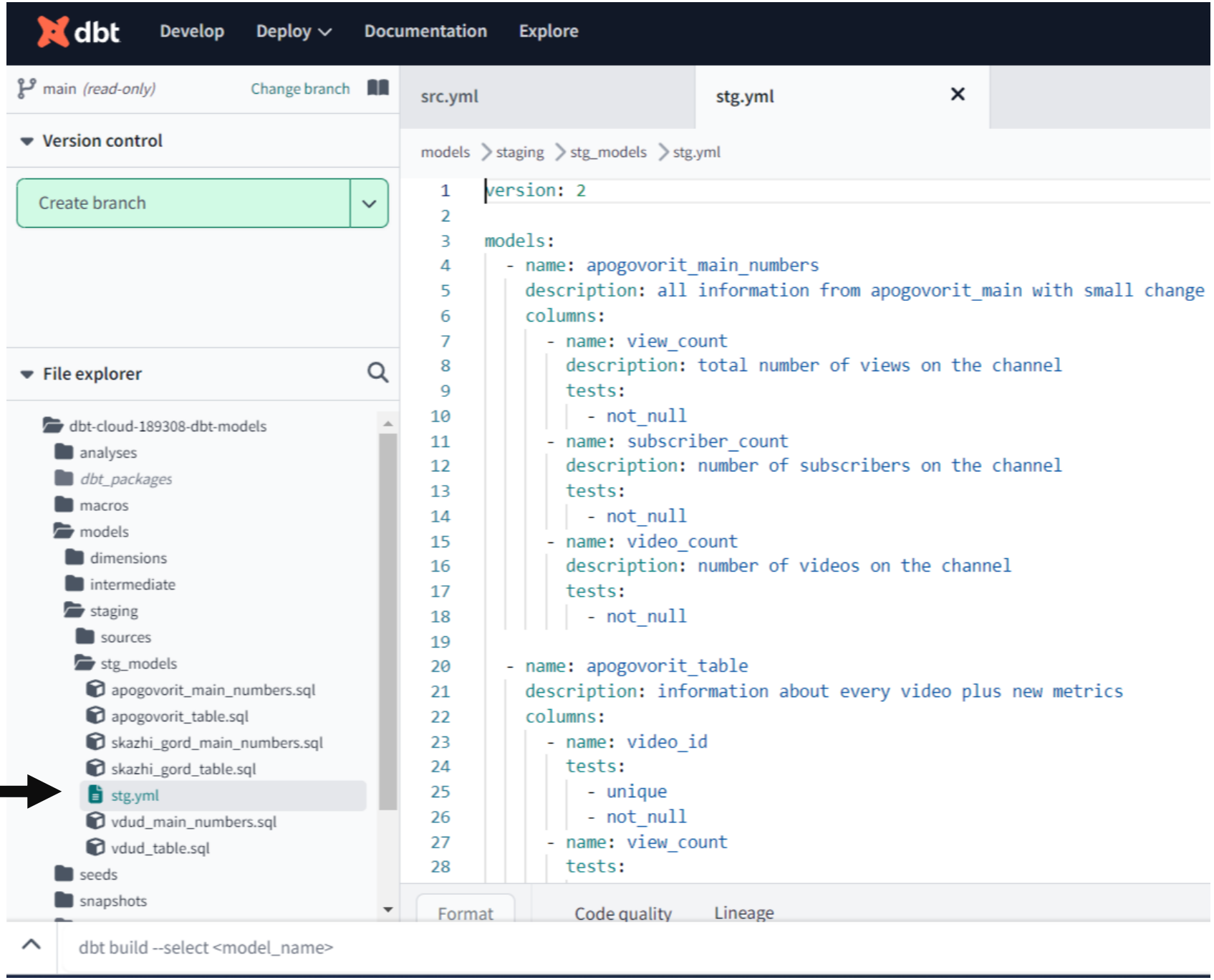
В папке **staging** 2 папки:

sources – здесь в файле src.yml описано подключение к источнику данных (Big Query) и все подробности о данных. Благодаря файлам .yml документация проекта формируется автоматически. Поэтому такие файлы находятся у меня в каждой папке.

stg_models – здесь в файлах .sql содержатся первые dbt-модели, а также есть и yml-файл

---

* все упоминаемые авторы youtube-каналов (Дудь, Шихман, Гордеева) - признаны иноагентами

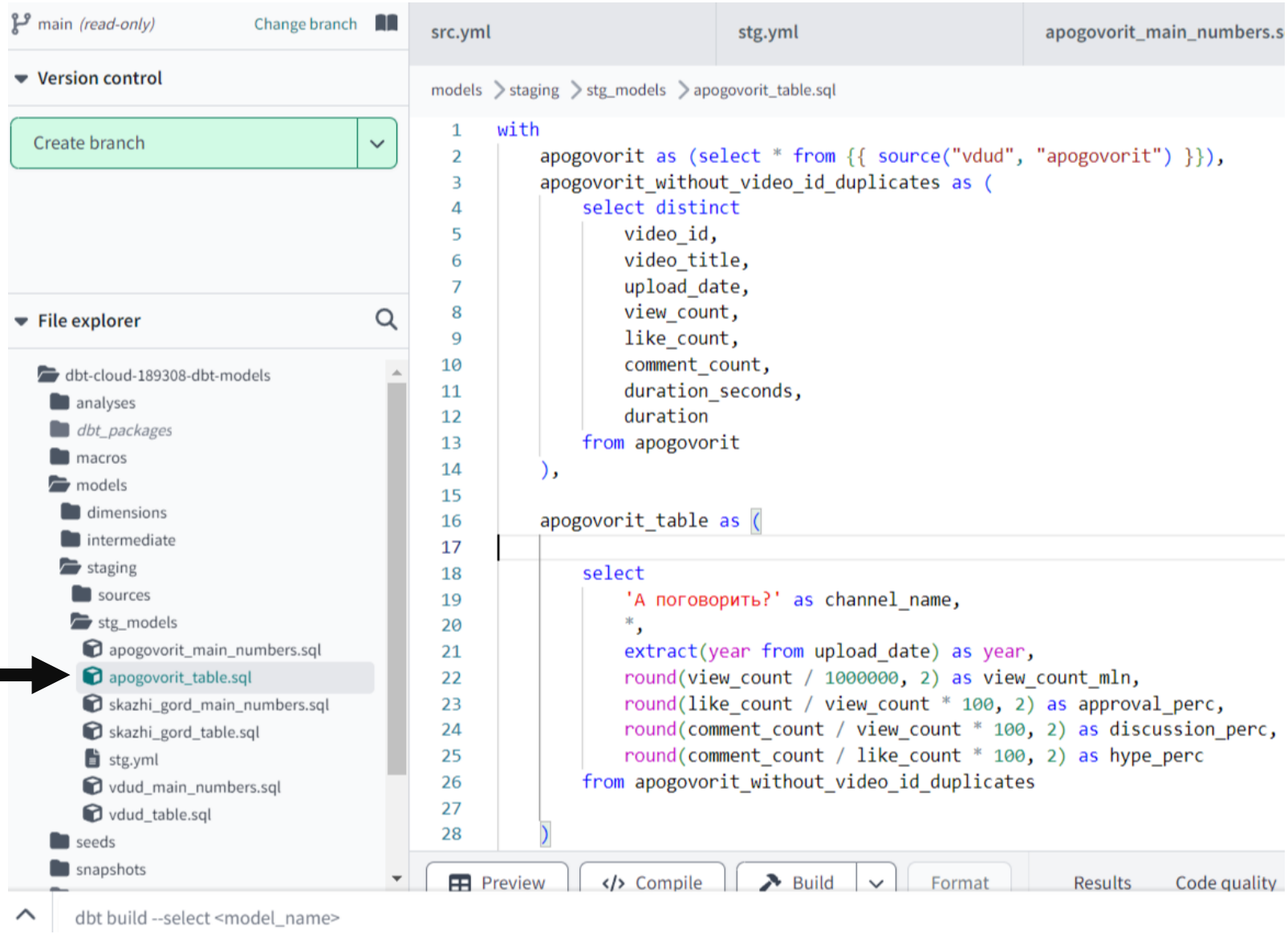


В файле stg.yml дано подробное описание самих моделей (таблиц):
- название
- описание
- колонки
    - название
    - описание
    - тесты

Таким образом, файлы .yml помогают не только создавать и документировать структуру проекта, но и очень простым способом сразу добавлять типовые тесты (unique, not_null)

---

* все упоминаемые авторы youtube-каналов (Дудь, Шихман, Гордеева) - признаны иноагентами

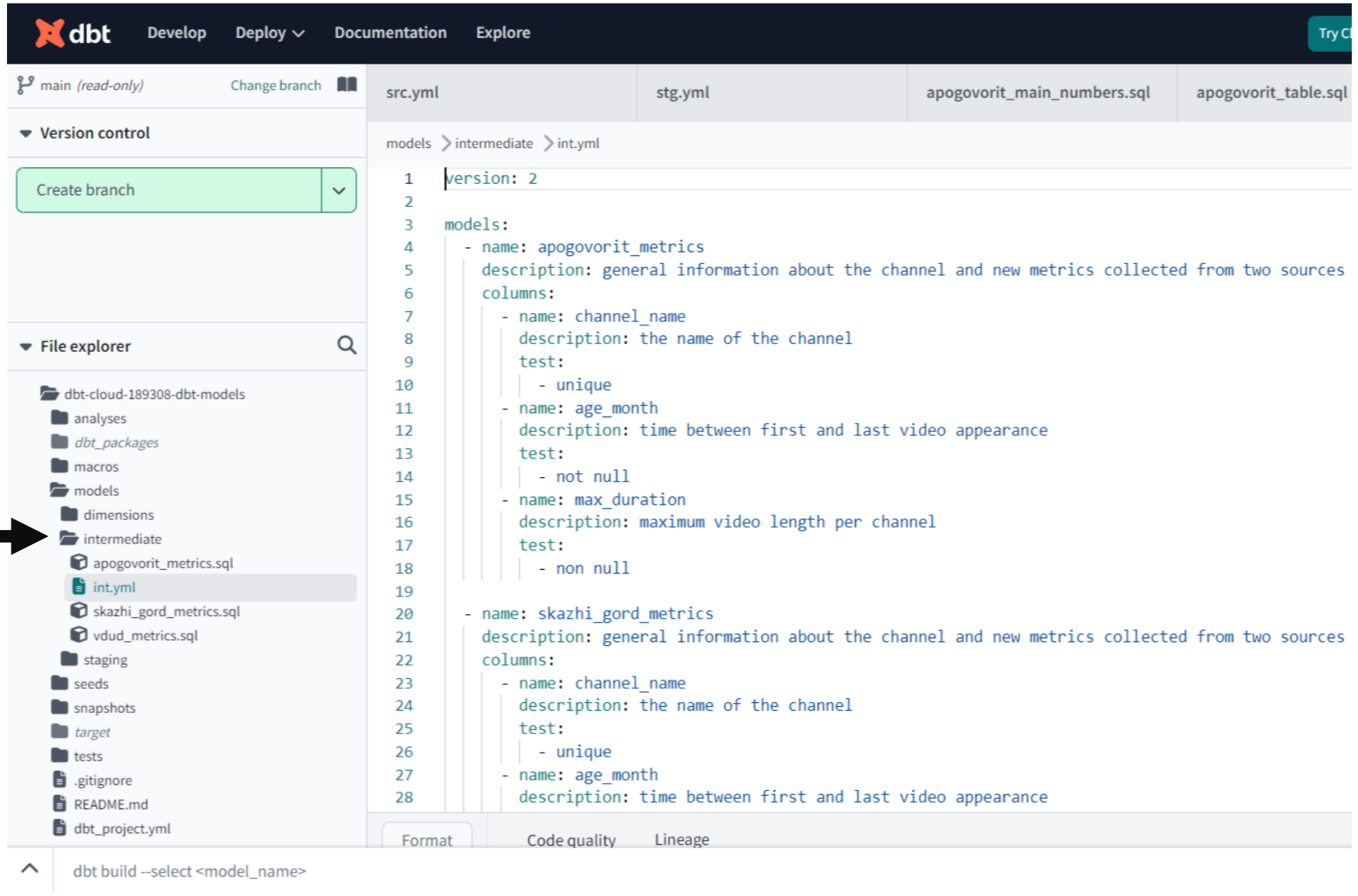


Если посмотреть какой-то sql-файл, то внутри мы увидим вполне привычные sql-выражения, но кроме этого и внутренние ссылки на другие шаги проекта.

Эти ссылки оформляются при помощи двойных фигурных скобок

---

* все упоминаемые авторы youtube-каналов (Дудь, Шихман, Гордеева) - признаны иноагентами

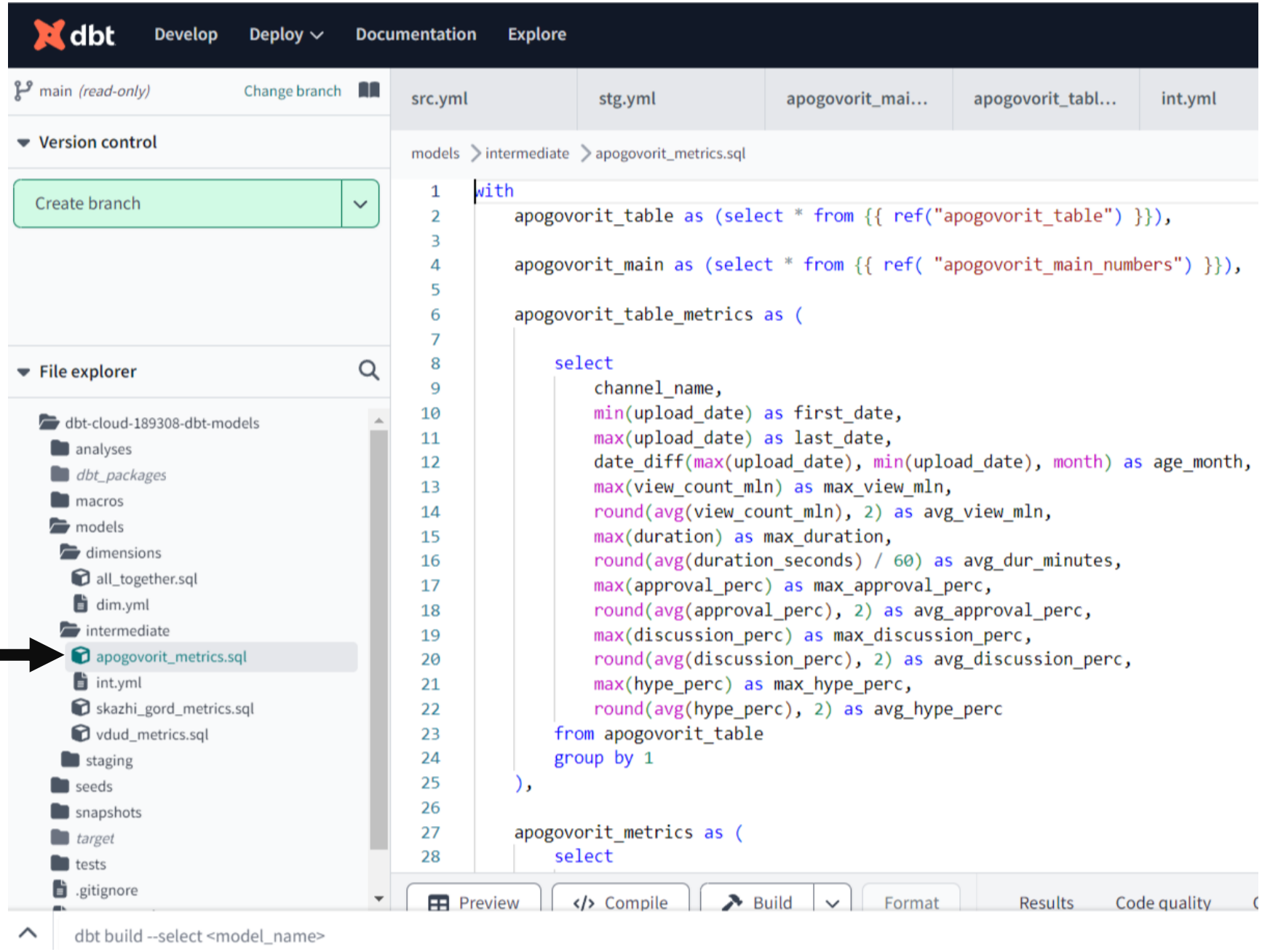


Следующая после staging папка – **intermediate**

здесь также есть yml-файл с описанием моделей, их колонок, тестов и т.п.

---

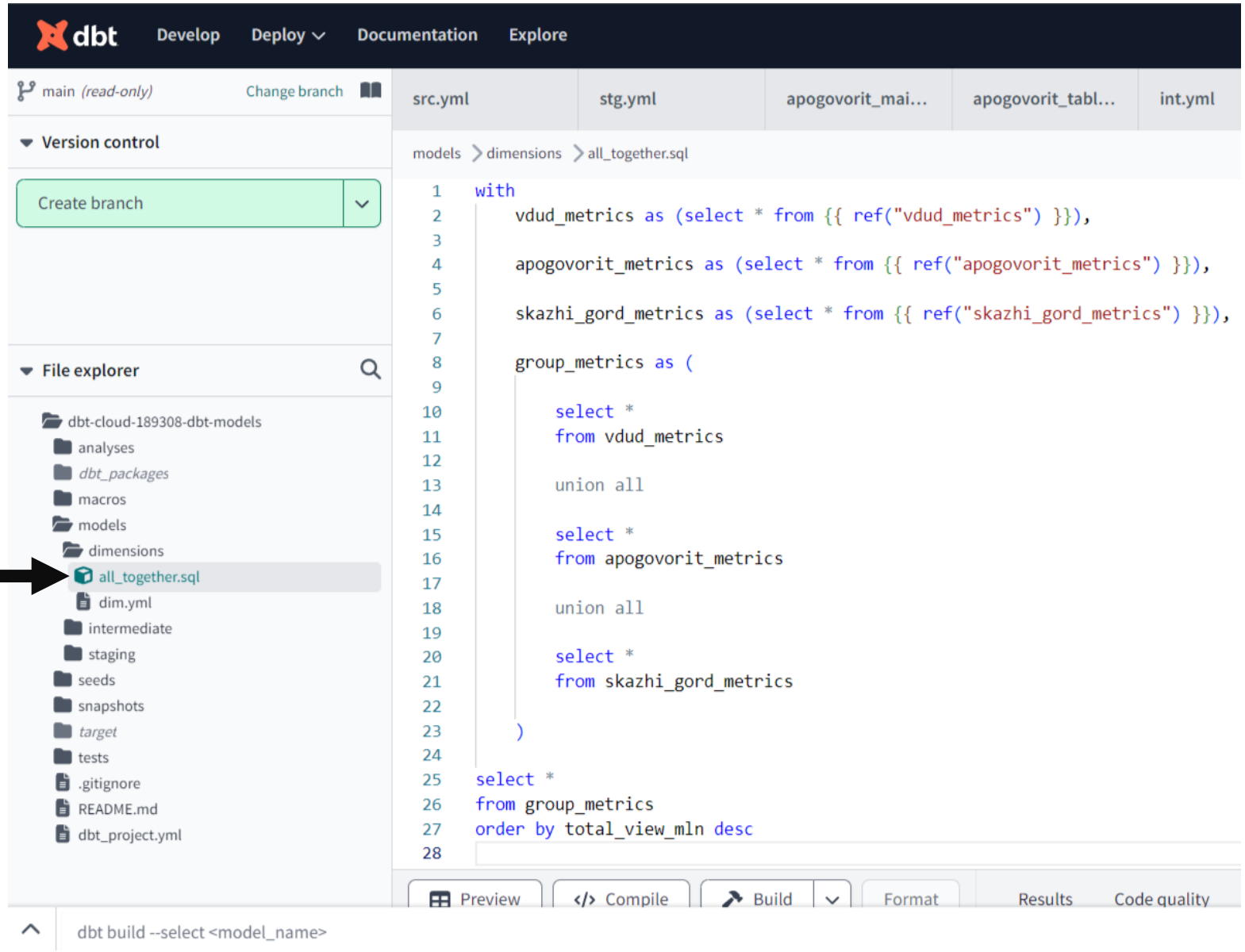
* все упоминаемые авторы youtube-каналов (Дудь, Шихман, Гордеева) - признаны иноагентами



На этом шаге происходят различные расчёты метрик

---

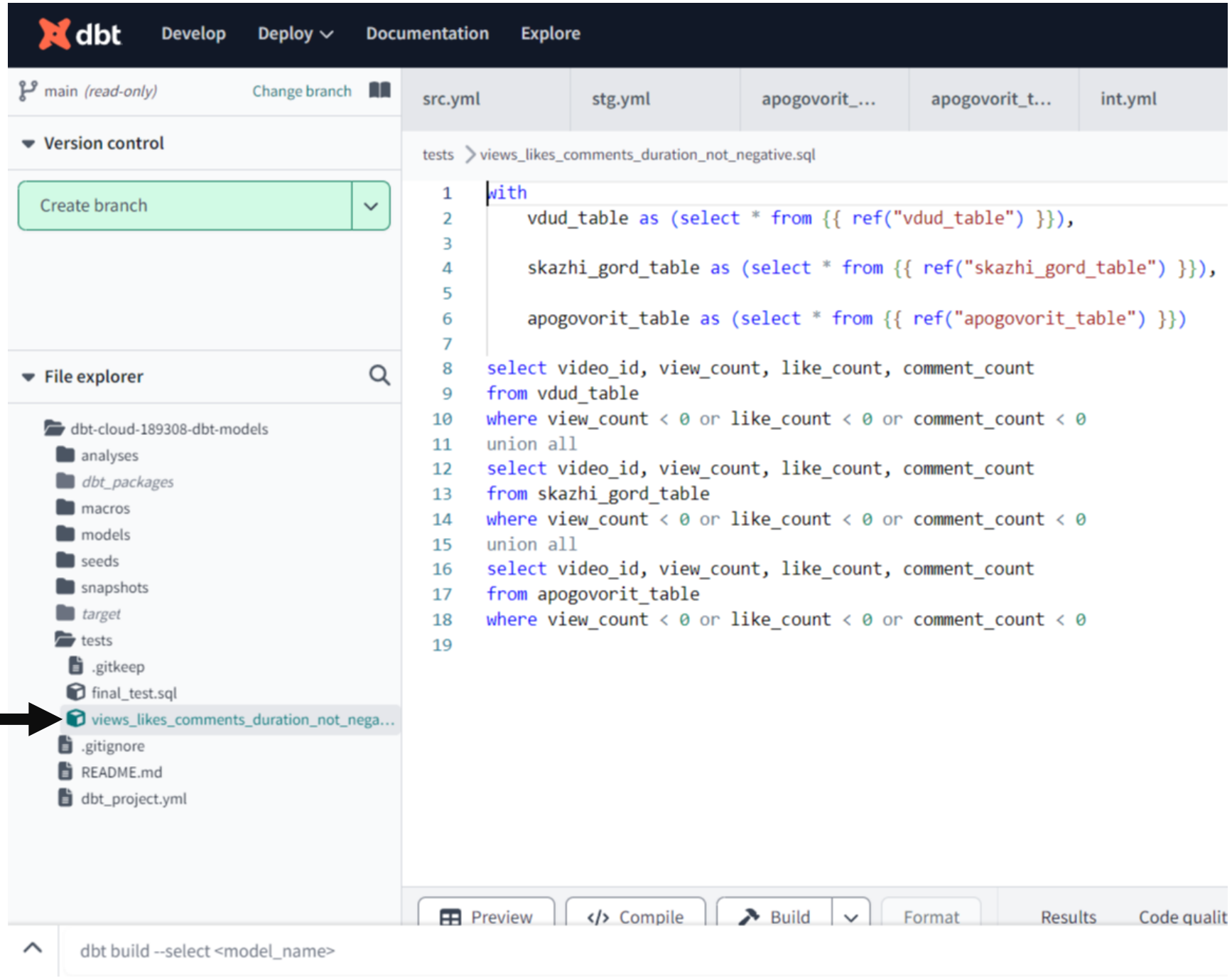
* все упоминаемые авторы youtube-каналов (Дудь, Шихман, Гордеева) - признаны иноагентами



На последнем шаге происходит объединение данных из разных источников

---

* все упоминаемые авторы youtube-каналов (Дудь, Шихман, Гордеева) - признаны иноагентами

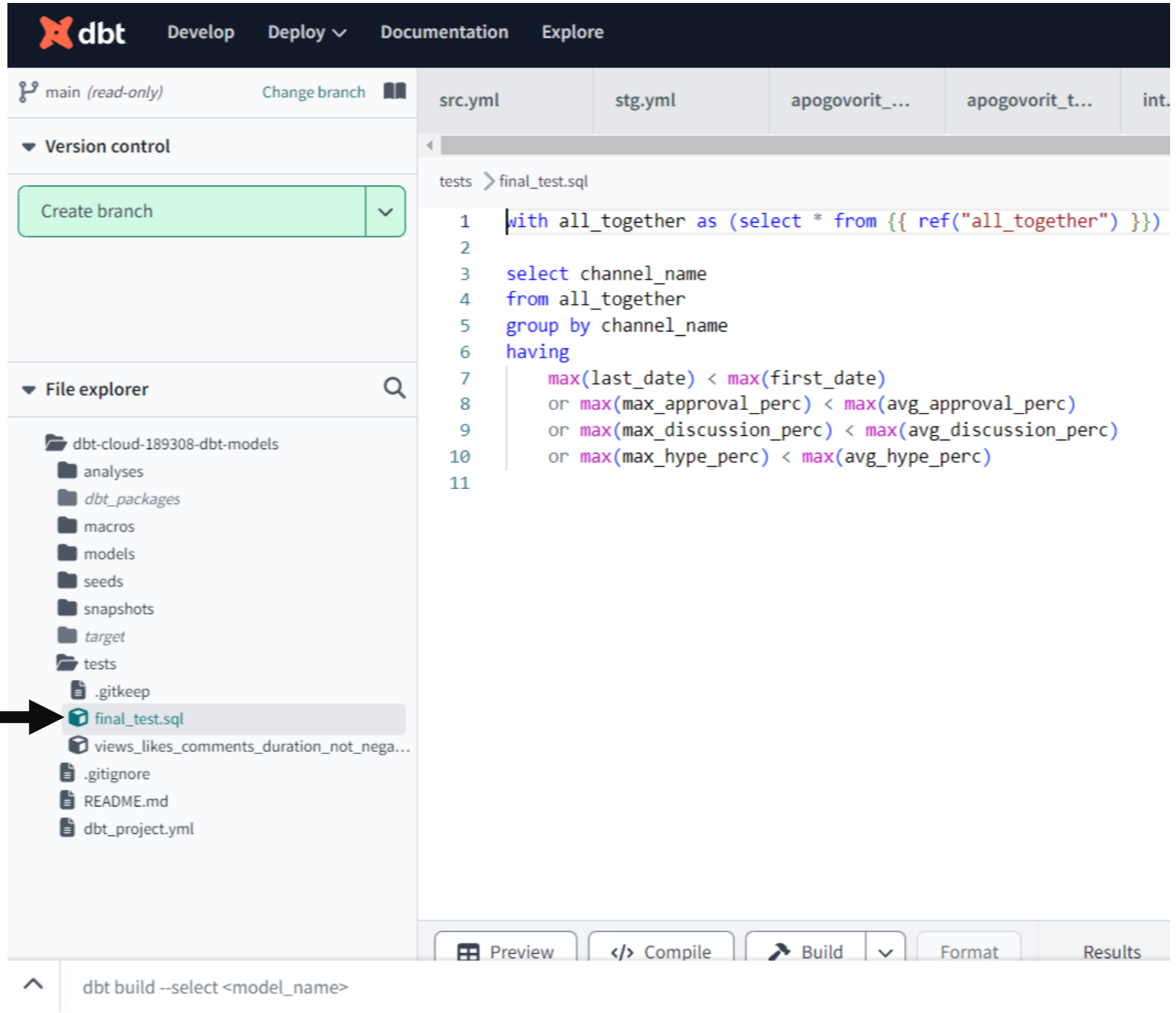


Кроме типовых тестов (not_null, unique) можно создать свои собственные тесты – они находятся в папке **tests**.

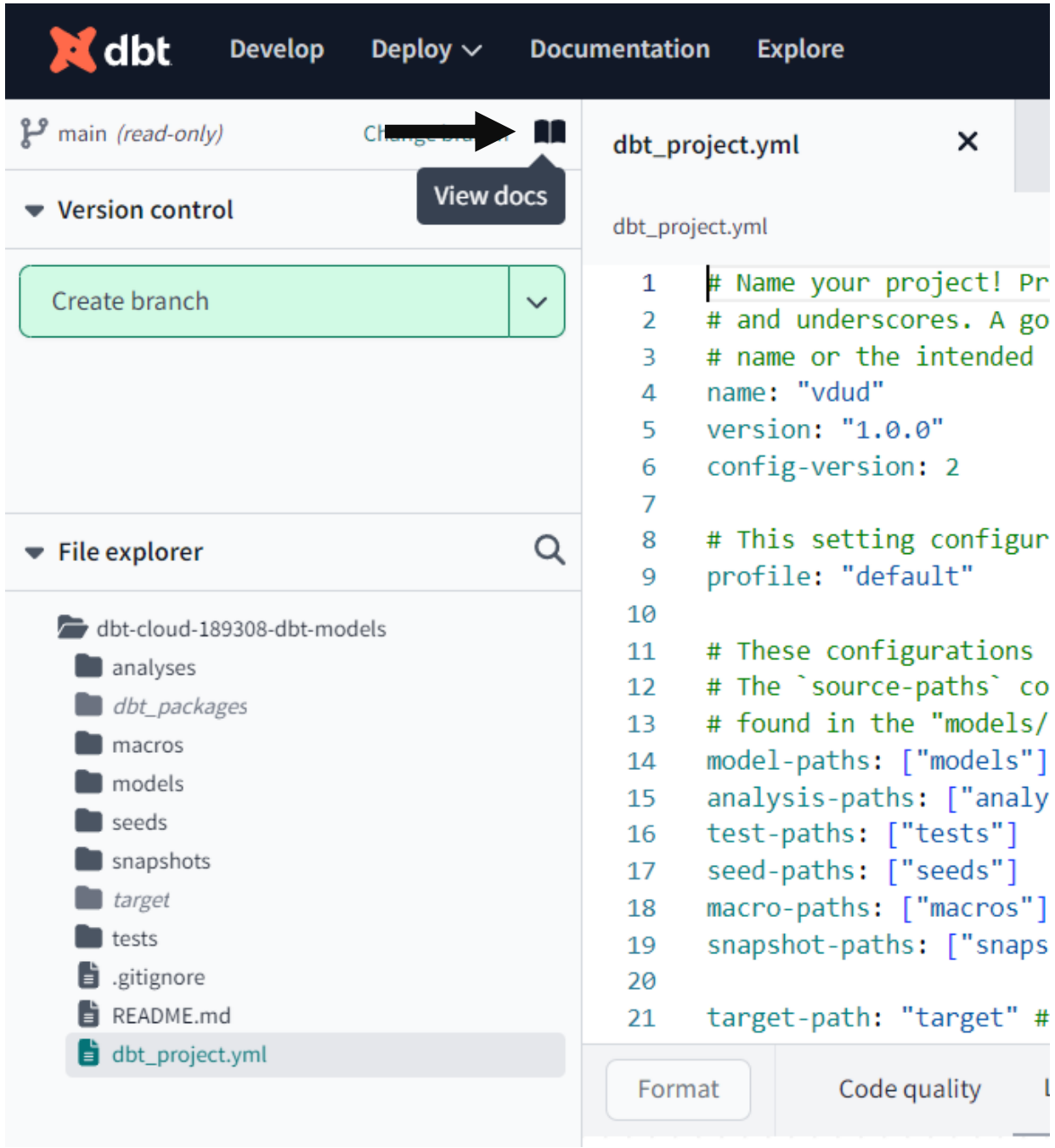
Например, я создала тест на выявление отрицательных значений по количеству
- просмотров,
- лайков,
- комментариев.

---

* все упоминаемые авторы youtube-каналов (Дудь, Шихман, Гордеева) - признаны иноагентами



Также в моём проекте есть ещё один собственный тест на определение ошибочных ситуаций – когда максимальные значения по каналу меньше средних значений по тому же каналу (т.е. это тест на адекватность расчётов метрик)

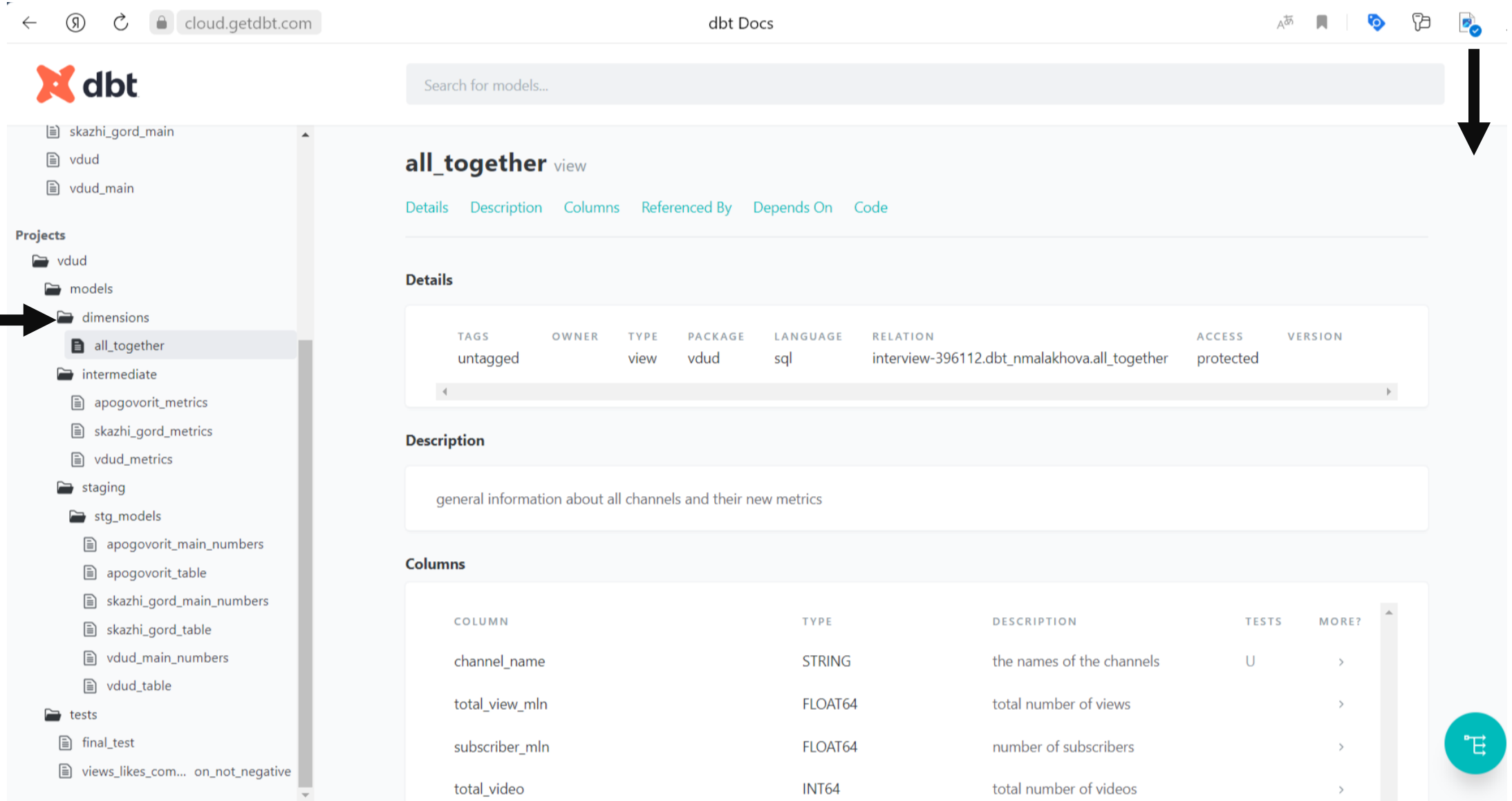


Теперь перейдём к документации проекта.

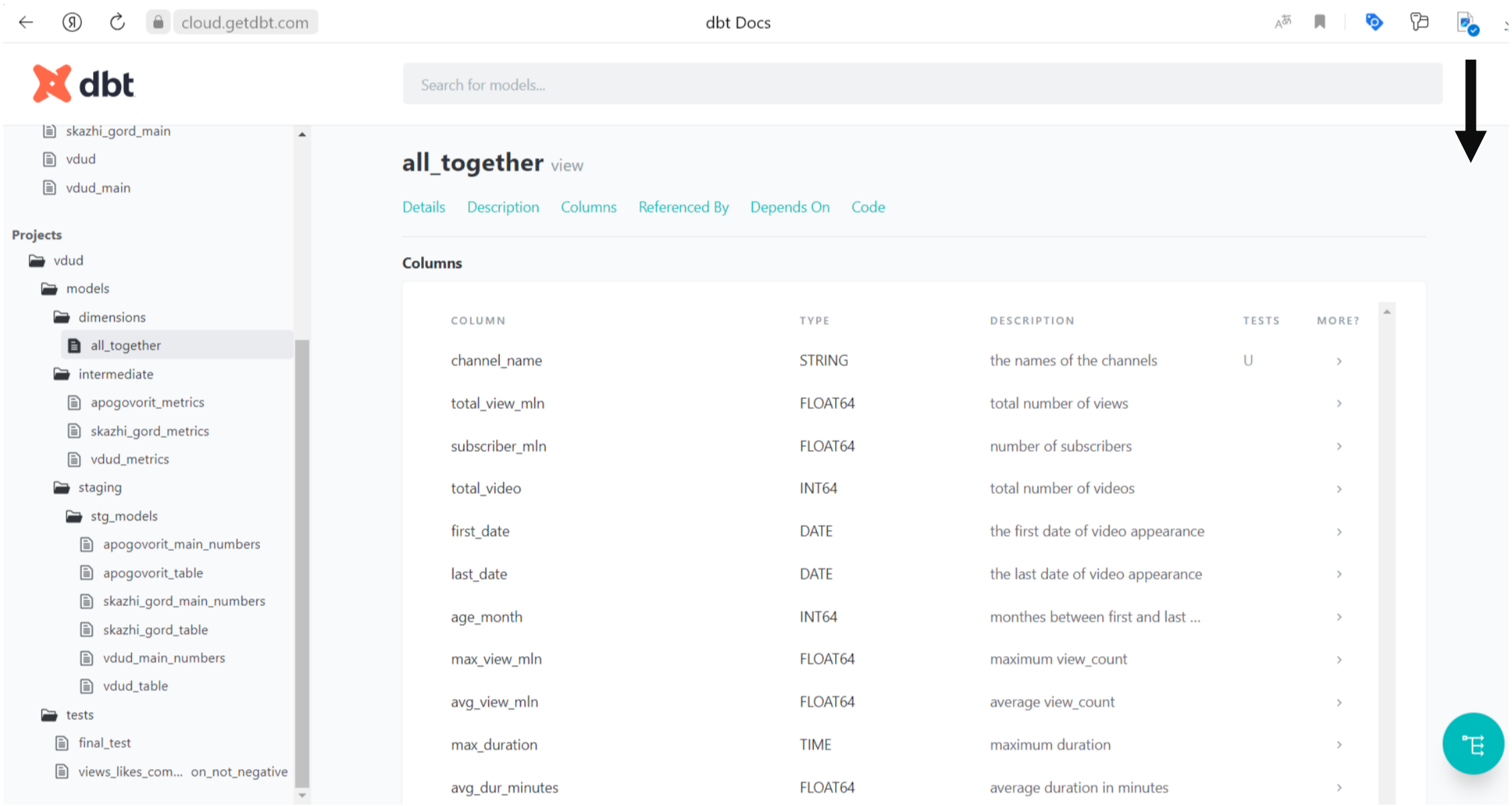
На неё можно перейти при помощи нажатия на иконку с раскрытой книгой.

Напомню, dbt Docs формируются автоматически, если в каждой папке своего проекта вы создаёте и заполняете yml-файлы.

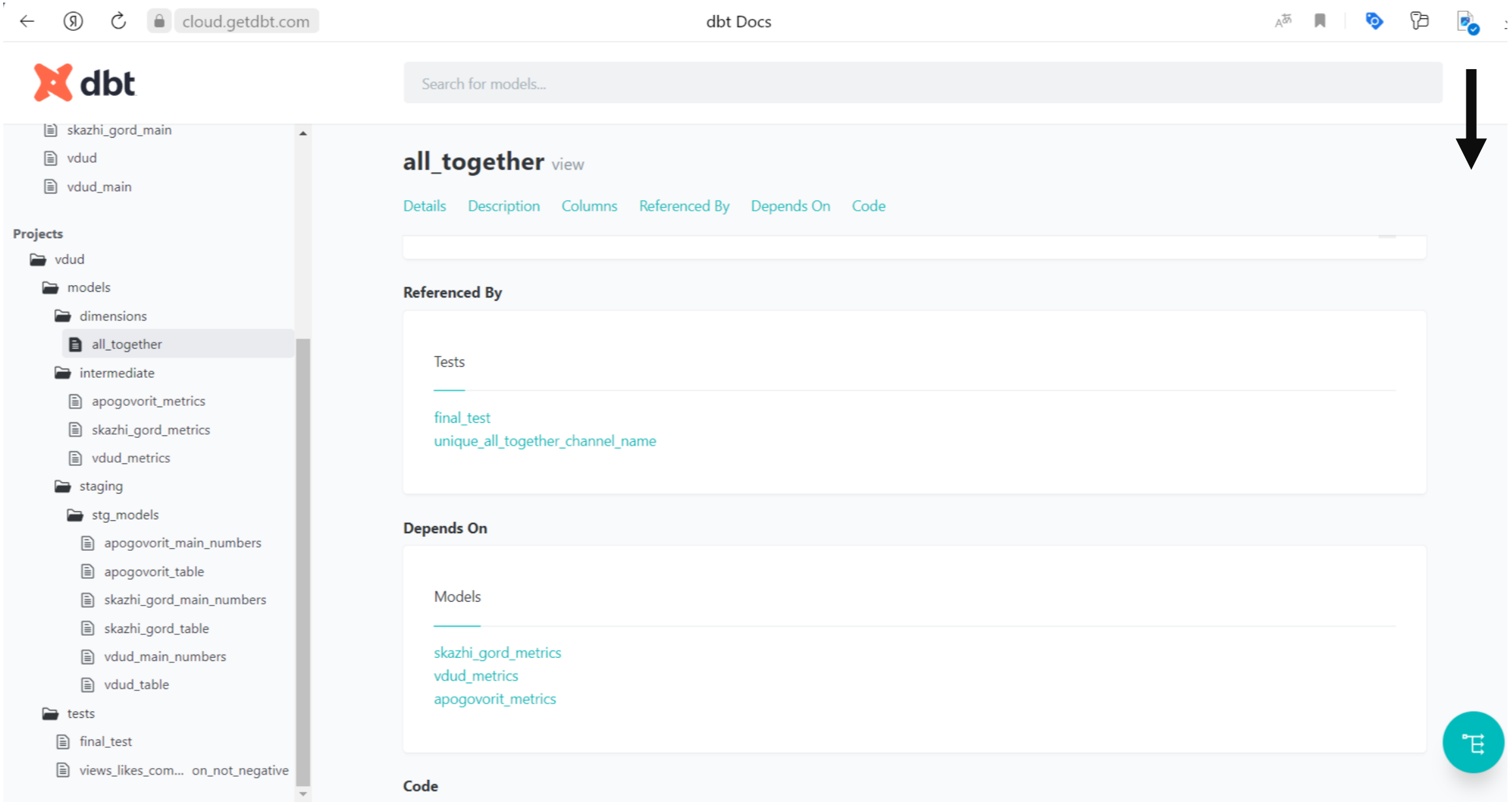
При открытии dbt Docs мы видим много всего. Для примера я открыла финальную модель all_together. Документация по модели начинается с деталей, описания, (далее для просмотра спускаемся вниз)…



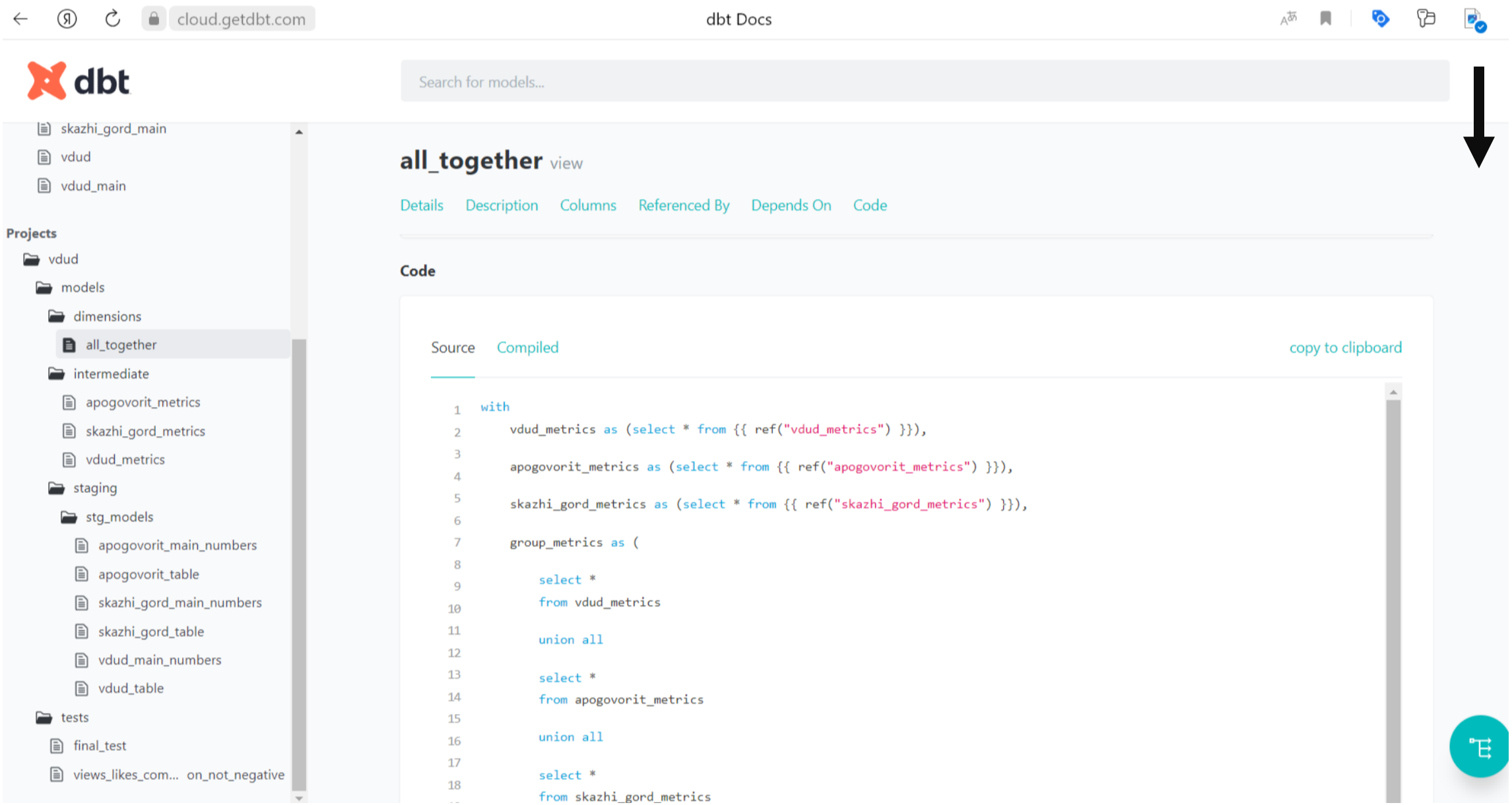
… далее мы видим колонки с типами данных и кратким описанием каждой…



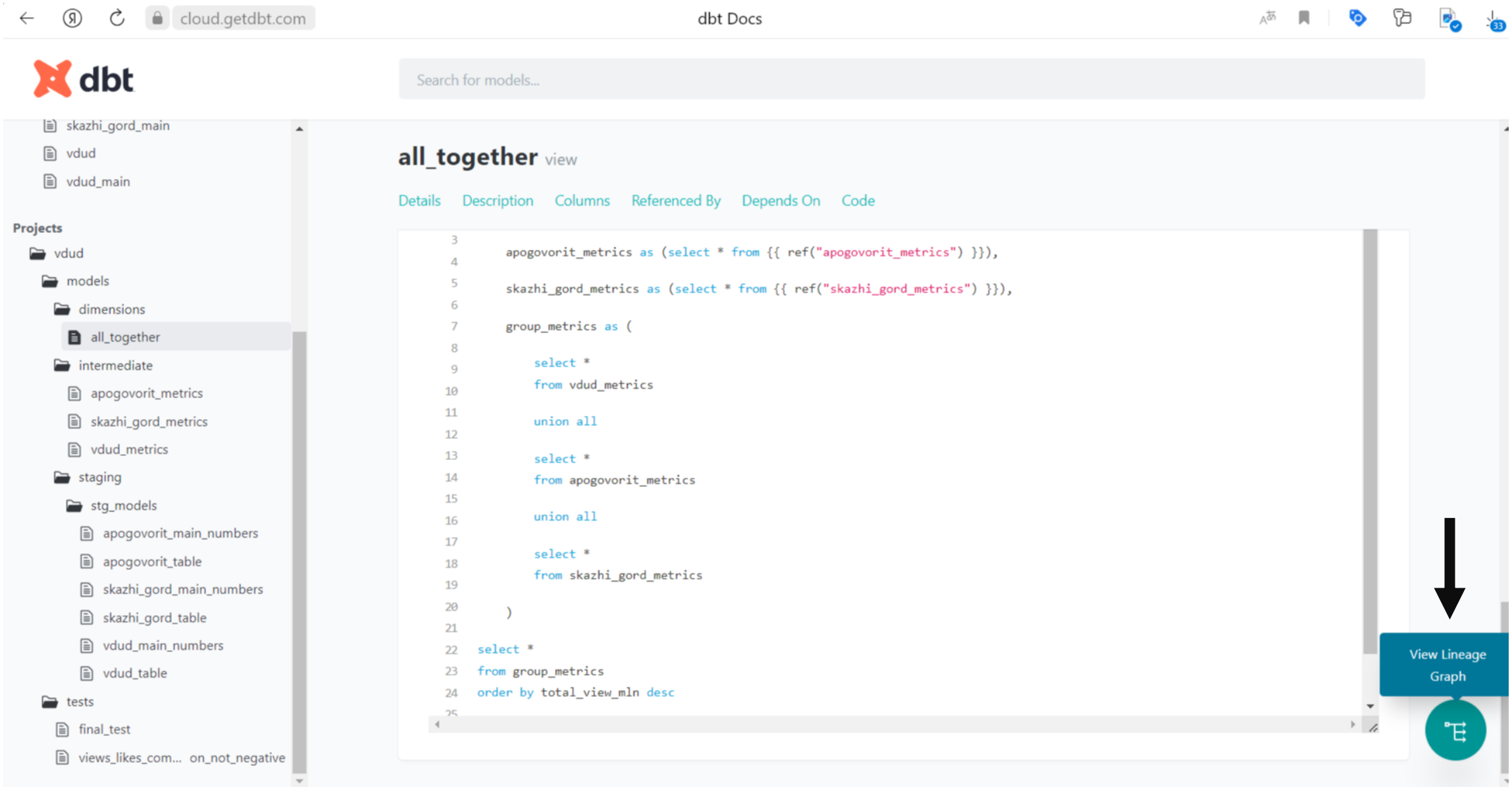
...ссылки, зависимости, ...



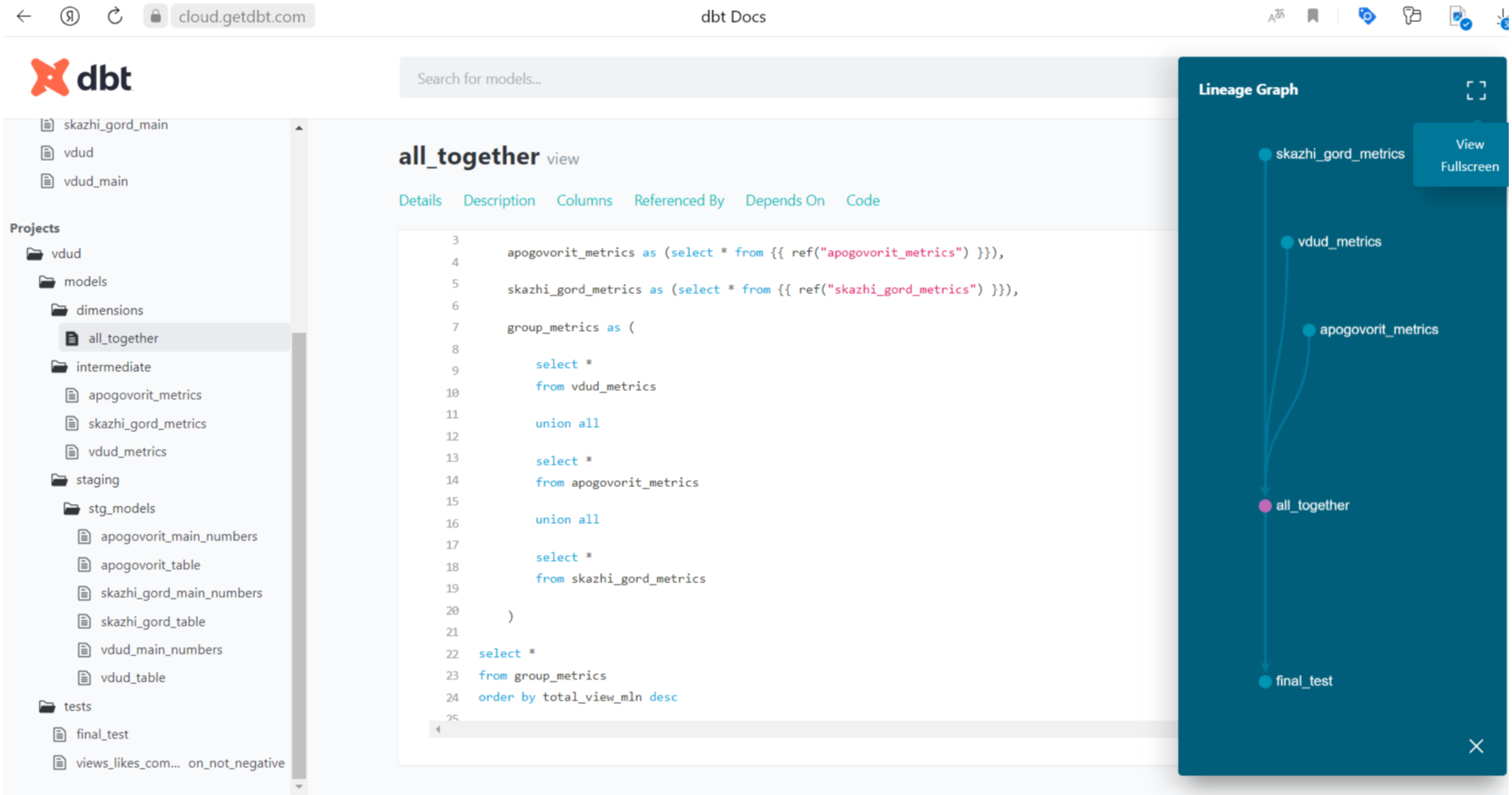
… сам код dbt-модели, …



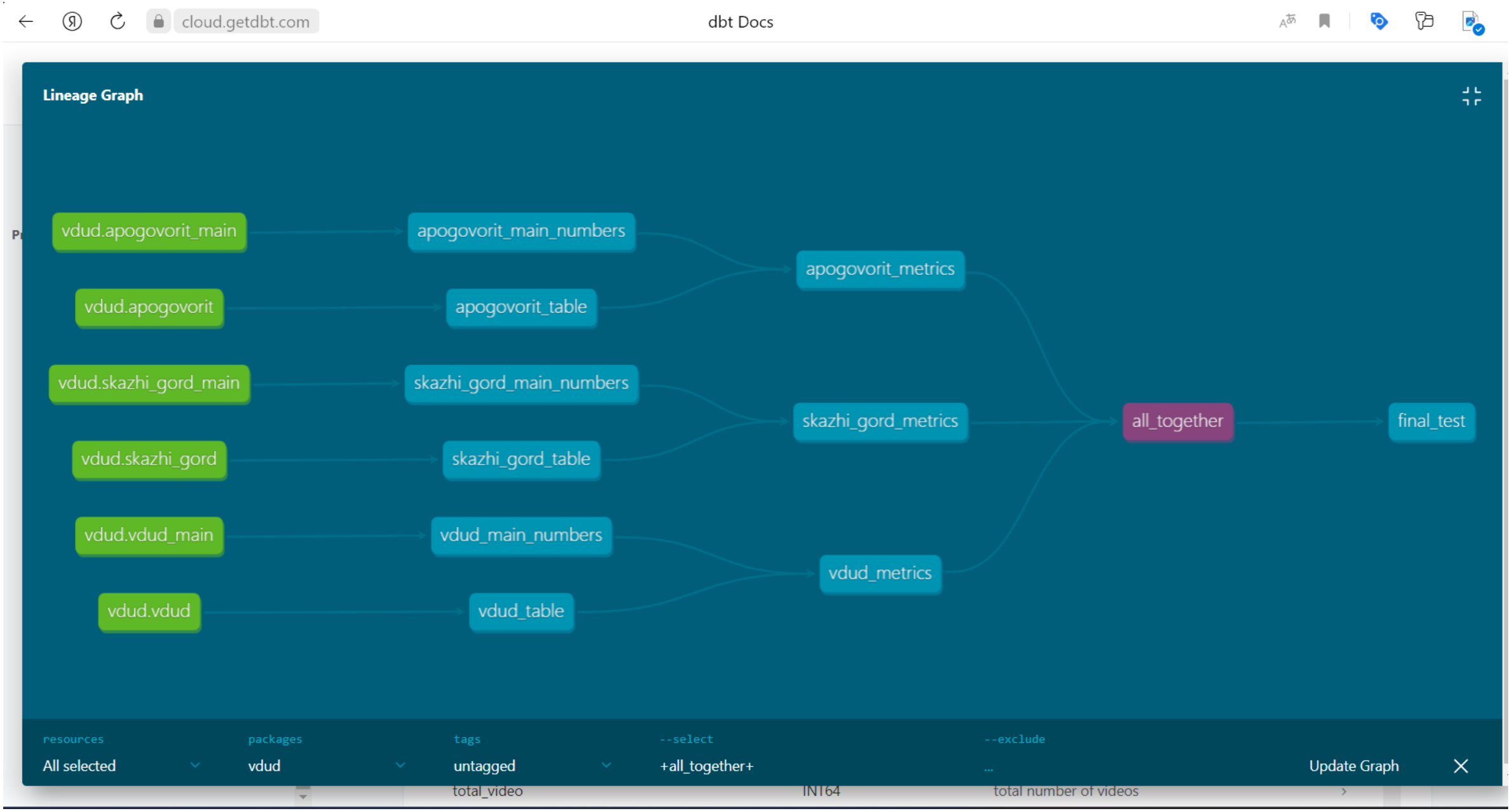
… и, наконец, моя любимая часть – граф – доступен при нажатии на кнопочку снизу



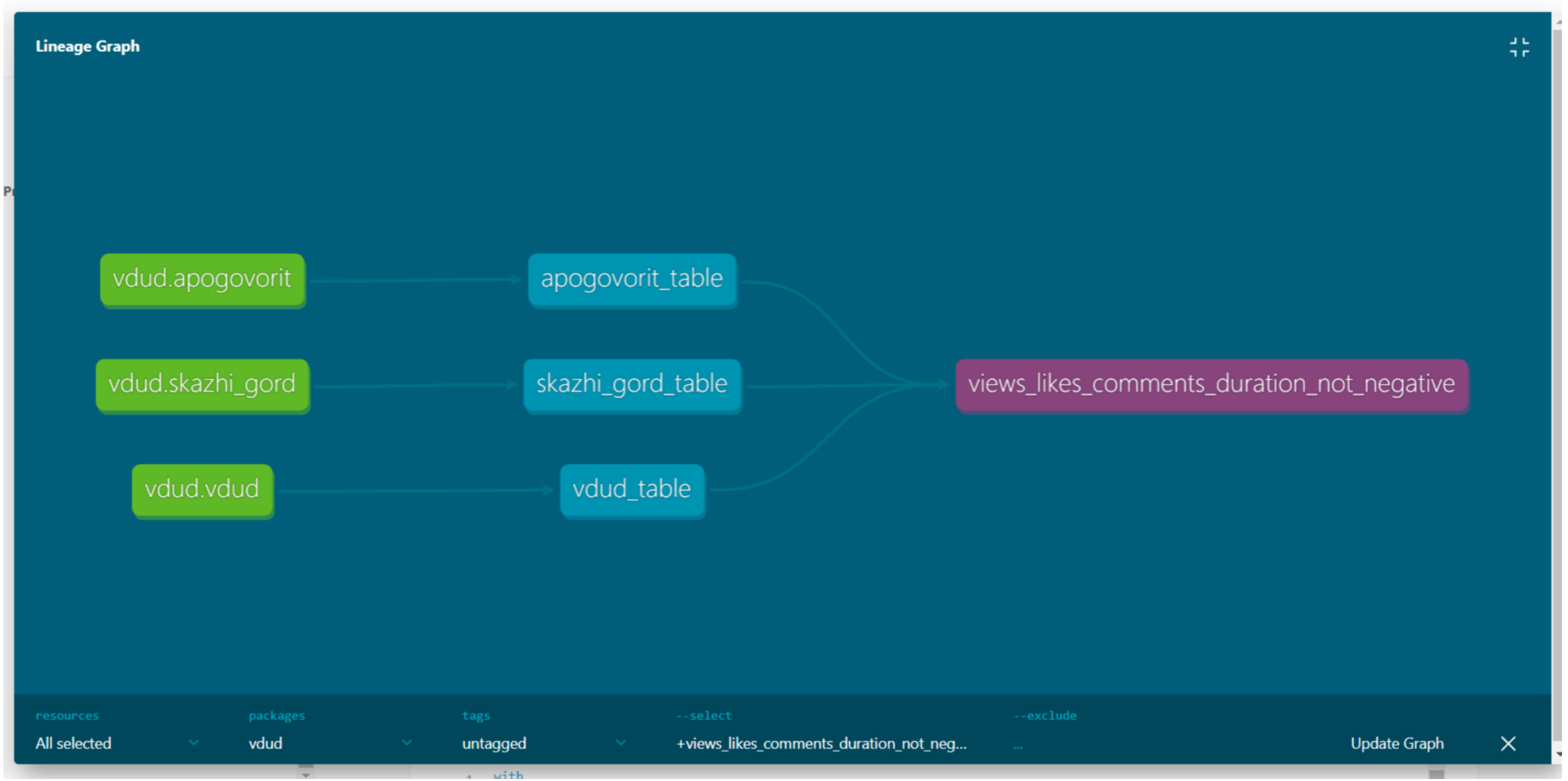
Граф показан в виде визуальной схемы зависимостей моделей (здесь видны соседние шаги для выбранной модели)



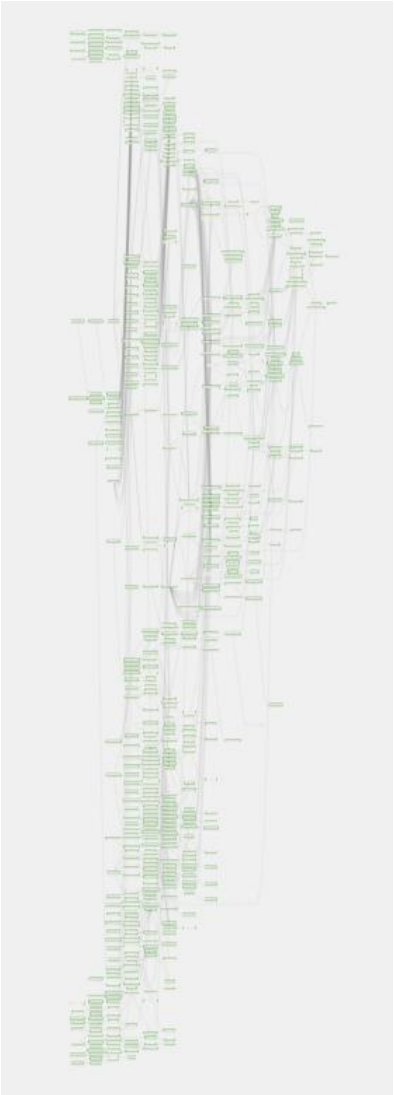
При желании граф можно раскрыть целиком



Кстати, графы доступны не только для моделей, но и для тестов
(views_likes_comments_duration_not_negative – это мой первый собственный тест)



Вот такой получился у меня проект.
Он находится по ссылке: https://cloud.getdbt.com/develop/189308/projects/276942



В заключение хотелось бы добавить, что dbt плотно используется у меня на работе. Доказательство тому – граф из Airflow с dbt_dag на картинке слева. Простыми словами это схема взаимосвязей dbt-моделей проекта.

Так что в моей рабочей жизни dbt – уже далеко не fancy инструмент ☺

Больше интересного о моём пути в аналитике данных – в моём телеграм-канале «Дневник аналитика»: https://t.me/diary_musician_analyst

Welcome!