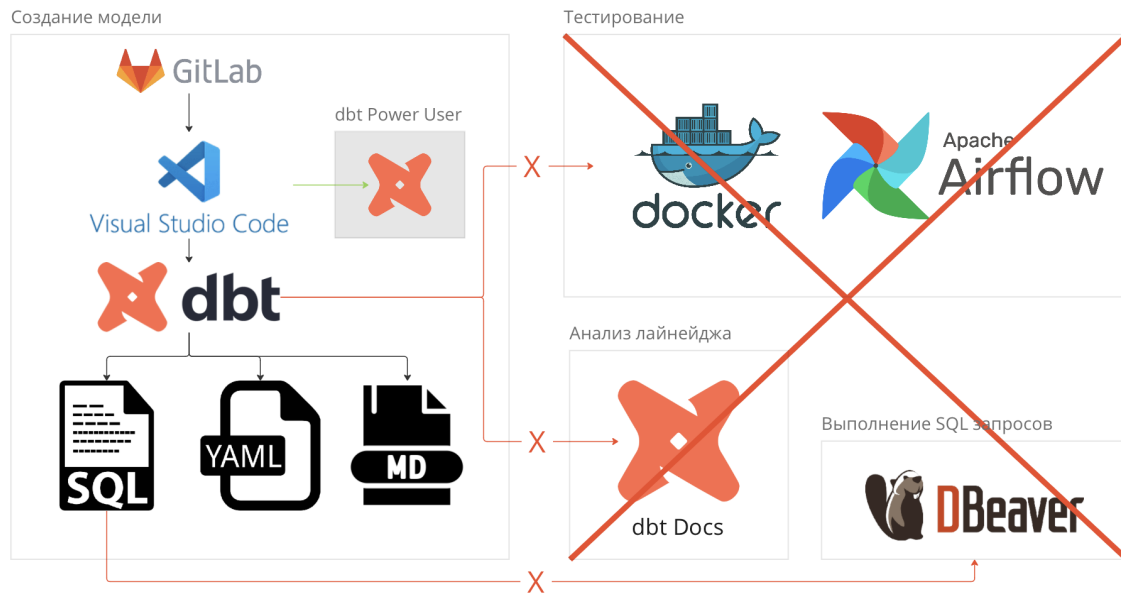


# dbt Power User

[blocked URL](#) **dbt Power User** - это расширение для **VSCode**, которое позволяет работать с dbt-проектом, экономя наше время и нервы.



- Основные преимущества
- Видео-демонстрация
- Установка и настройка
  - Подготовка системы (macOS)
  - Подготовка VS Code
- Тестирование моделей: до и после
- Как это работает
- Примеры
  - Execute dbt SQL
  - Build dbt Model
  - Переход по ссылкам
  - Нюансы с TMP
  - YAML Red Hat
  - dbt Docs
- Полезные ссылки

## Основные преимущества

1. **Тестирование dbt моделей без докера.** Не поднимая локальный Airflow, можно запустить свою модель из **VS Code** и получить аналогичный результат, который получили бы в результате выполнения дага;
2. **Режим одного окна для SQL запросов.** Оставаясь в **VS Code**, можно выполнять SQL-скрипты, при этом компиляция `{{ source }}` и `{{ ref }}` ссылок производится автоматически;
3. **Встроенный лайнейдж.** Находясь в SQL-скрипте, можно увидеть все родительские и дочерние модели текущего скрипта, а также мгновенно переместиться к любой из них;
4. **Удобная навигация по модели.** Находясь в SQL-скрипте, по одному клику можно переместиться к \*.yaml-конфигу, а оттуда к \*.md-файлу с описанием, не прибегая к помощи проводника;
5. **Автоподстановка кода.** Набирая код, не обязательно набирать весь текст вручную, например структура `{{ ref ("", "") }}` подставляется автоматически (как и нейминг таблиц);

Для тестирования своих MRов рекомендуется использовать утилиту: [dbt\\_reviewer](#), так как она настроена на наш проект и требует меньше ручных манипуляций в части поддержания актуальности своих локальных конфигов.

 **Видео-демонстрация**

**Установка и настройка**

## Подготовка системы (macOS)

0. В системе должен быть установлен Python версии 3.9 и ниже. При установке не забыть прожать галочку о добавлении интерпретера в PATH.

Уедиться, есть ли путь до Python 3.9 в PATH можно вы полнив команду в терминале

terminal (mac)

```
echo $PATH;
```

Если пути до нужного питона нет, его необходимо прописать

terminal (mac)

```
sudo nano /etc/paths;
```

Например мой путь: /Users/kustov.ae/Library/Python/3.9/bin (**не** /usr/bin/python3)

После добавления пути в PATH перезагружаем терминал

Резуьтат, которого мы хотим добиться - **pip** запускается из терминала (не **pip3**) и к библиотеке Python (например **dbt**), можно обратиться напрямую из терминала

Если команда **pip** работает, но не запускается либа из терминала, например видим **dbt: command not found**, то нужно убедиться, что в PATH прописан именно путь до системного интерпретера

Команда **pip install --upgrade pip** выдаст WARNING и подсветит путь, который должен быть прописан в PATH.

Если ничего не выходит, настроить VENV и запускать из под него

1. Устанавливаем dbt через терминал (актуальные версии пакетов текущего проекта можно посмотреть в файле [requirements.txt](#))

Terminal

```
pip install dbt-core==1.1.1 dbt-postgres==1.1.1
```

- если команда **pip** не работает, значит у нас что-то неверно настроено. Разбираемся с PATH. **pip3** тут использовать нельзя

2. Создаем файлы, необходимые для локального запуска dbt проекта (либо просто используем [dbt\\_reviewer](#))

Terminal

```
mkdir ~/.dbt; touch ~/.dbt/dbt_project.yml ~/.dbt/profiles.yml;
touch ~/.zshenv; echo export DBT_PROFILES_DIR=~/.dbt" >> ~/.zshenv;
```

3. Прописываем внутри файлов набор конфигураций (в ~/.dbt/profiles.yml необходимо вставить свои креды от GP DEV)(либо просто используем [dbt\\_reviewer](#))

## ~/dbt/profiles.yml

```
config:
  debug: true
  send_anonymous_usage_stats: false
  use_colors: true
  partial_parse: true

dbt_dags:
  outputs:

    dev:
      type: postgres
      threads: 4
      host: "c-c9qfrnqkuod6v7ive03r.rw.mdb.yandexcloud.net"
      port: 5432
      user: "<login>"
      pass: "<password>"
      dbname: "dwh"
      schema: "public"

  target: dev
```

## ~/dbt/dbt\_project.yml

```
name: 'dbt_dags'
version: '1.0.0'
config-version: 2

profile: 'dbt_dags'

model-paths: ["models"]
analysis-paths: ["analyses"]
test-paths: ["tests"]
seed-paths: ["seeds"]
macro-paths: ["macros"]
snapshot-paths: ["snapshots"]
docs-paths: ["docs"]

target-path: "target" # directory which will store compiled SQL files
clean-targets:         # directories to be removed by `dbt_dags clean`
  - "target"
  - "dbt_packages"

on-run-start:
  - "{{ log_table_status(operation_status('START')) }}"

on-run-end:
  - "{{ drop_temp_tables(results) }}"

vars:
  default_max_dttm: '9999-12-31 00:00:00.000'
  pxf_rowcount_accuracy: 0.999
  process_id: 0
  load_dttm: '2024-01-01 00:00:00.000'
  process_type: dbt
  dag_id: dbt_power_user_test
  task_execution_timeout: 100
  meta_schema: "meta"
  meta_log_schema: "meta_log"
  main_table_name: "main_table_name"
  ext_vars:
    ext_tbl_location: ""
    ext_max_id: 0

models:
```

```

+pre-hook:
  sql: "{{ pre_hook_general() }}"
  transaction: false
+post-hook:
  sql: "{{ post_hook_general() }}"
dbt_dags:
+materialized: table
scripts:
  detail:
    +docs:
      node_color: "Green"
  preaggregate:
    +docs:
      node_color: "Red"
  datamart:
    +docs:
      node_color: "Blue"
  report:
    +docs:
      node_color: "BlueViolet"
  cte:
    +materialized: ephemeral
    +docs:
      node_color: "LightSkyBlue"
  tmp:
    +schema: tmp
    +materialized: temp_table_materialization
    +docs:
      show: false
    +meta:
      temporary: true
  vw:
    +post-hook:
      sql: "{{ grant_table(this) }}"
      transaction: true

```

4. Проверяем, что все настроено верно - ожидаем увидеть «All checks passed!»

#### Terminal

```
cd ~/.dbt; dbt debug;
```

В некотором случае, который сейчас не поддается дебагу, при установленном Python 3.9 и Python 3.11 может возникнуть ситуация, что команда dbt debug обращается к Python 3.11, даже если в \$PATH указан путь до Python 3.9.

Текущий вариант решения: в нужной папке поднять виртуальное окружение (VIRTUALENV / VENV) с нужной версией питона и установить DBT-CORE и DBT-POSTGRES внутри окружения

#### Terminal

```

# pip python , pip3 python3
#
cd dwh-dags
# virtualenv, ,
pip install -U virtualenv
# venv ( venv)
python -m venv .venv
# (venv)
source .venv/bin/activate
#
pip install dbt-core==1.1.1 dbt-postgres==1.1.1

```

Чтобы Git не видел папку .venv достаточно создать внутри файл .gitignore следующего содержания:

```
# (dwh-dags)
echo "*" > ../.venv/.gitignore
```



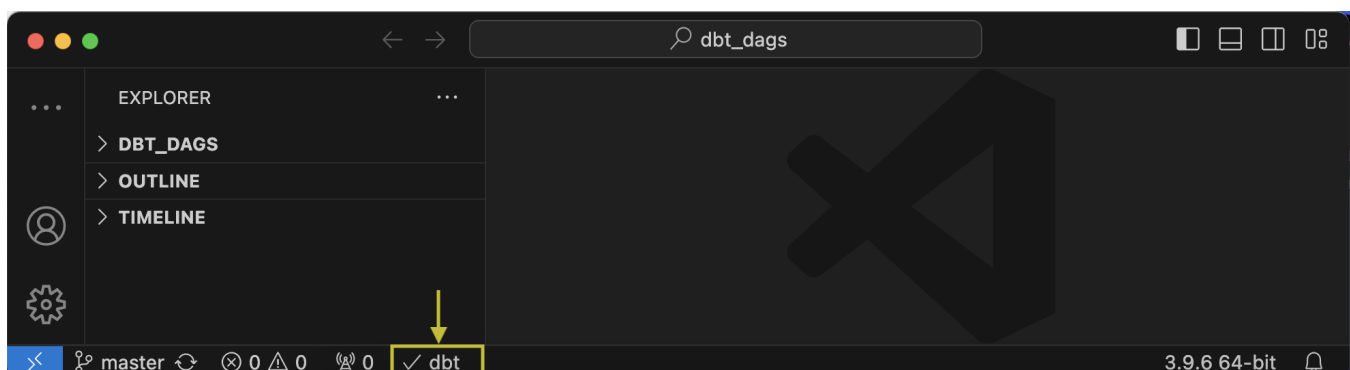
## Подготовка VS Code

1. [Устанавливаем расширение dbt Power User в VSCode](#) (⚠ версию 0.43.5, старше этой версии утилита некорректно работает с нашим dbt-project.yml)
2. Открываем конфиг VS Code: cmd()+shift()+P (>Preferences:Open User Settings (JSON)) и прописываем ряд дополнительных настроек:

~/Library/Application Support/Code/User/settings.json

```
{
  "files.associations": {
    "*.yml": "jinja-yaml",
    "*.yaml": "jinja-yaml",
    "*.sql": "jinja-sql",
  },
  "dbt.enableNewLineagePanel": true,
  "dbt.queryLimit": 10,
  "dbt.enableNewDocsPanel": true,
  "python.interpreter.infoVisibility": "always",
}
```

3. Открываем репозиторий dwh\_dags и на панели снизу ожидаем увидеть галочку напротив dbt:



# Тестирование моделей: до и после

Для того, чтобы dbt Power User заработал, необходимо временно подменить dbt\_dags/dbt\_project.yml на локальную версию

## Terminal

```
cp -R ~/.dbt/dbt_project.yml dbt_dags/dbt_project.yml;
```

По окончании тестирования **ОБЯЗАТЕЛЬНО** выполняем команду

## Terminal


```
git restore dbt_dags/dbt_project.yml;
```




**Важно** не забыть отменить изменения до коммита, т.к. файл не находится в .gitignore и является **критическим объектом**


## Как это работает

После установки на панели слева появится расширение dbt Power User. При открытии SQL-скрипта, можем увидеть следующие фичи:

1. **PARENT MODELS** - все родительские модели (переход по клику);
2. **CHILDREN MODELS** - все дочерние модели (переход по клику);
3. **DOCUMENTATION** - описание полей из \*.yml файла (переход в конфиг по клику);
4. **LINAGE VIEW** - визуализация связи текущего SQL с родительскими и дочерними моделями (переход по клику);

5.  - выполнение sql запроса и получение результата в **QUERY RESULTS**;

6.    - build / run / test dbt-модели;

7.  - компиляция sql-скрипта: заменяются все ref и source + подставляются все используемые cte.

Актуальную документацию программы можно [читать здесь](#)

The screenshot displays the dbt Power User interface. On the left, a sidebar shows a tree view of models categorized into CHILDREN MODELS, PARENT MODELS, and DOCUMENTATION. The main panel shows the SQL script for the `s_order` model. Below the script, the 'LINEAGE VIEW' tab is active, showing a diagram of data dependencies. The diagram illustrates that the `s_order` model (purple box) receives data from parent models: `tmp_orders_b2b`, `tmp_spree_addresses_phone_act`, `all_layers_params (meta)`, and `vw_spree_orders (vw_ods_inst)` (all in blue boxes). The `s_order` model then feeds into child models: `cac_shipment_tmp`, `p_adjust_sources_tmp`, `p_adjustment_cte`, and `p_appsflyer_logs` (all in orange boxes). A yellow arrow points from the 'LINEAGE VIEW' tab in the top right to the lineage diagram.

## Примеры

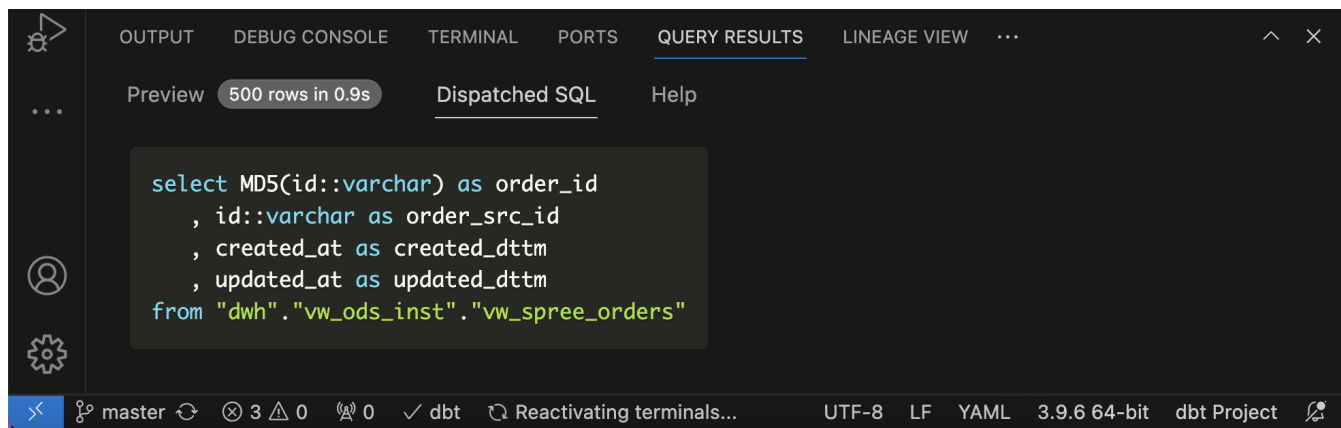
### Execute dbt SQL

При выполнении SQL-скрипта результат будет выведен в следующем виде:

The screenshot shows the dbt Power User interface with the `h_order.sql` script loaded. The 'QUERY RESULTS' tab is active, displaying a preview of 500 rows in 0.9s. The results are shown in a table format with columns: `order_id`, `order_src_id`, `created_dttm`, and `updated_dttm`. Buttons for 'Copy Results' and 'Download as CSV' are visible above the table.

order_id	order_src_id	created_dttm	updated_dttm
a924bfc9661a49cd0b2c878b015a8d79	92052164	2021-04-22T11:52:59	2023-06-06T18:28:49
f4c59248f341a396249f5c45fe595a34	178632208	2021-12-11T11:07:19	2023-06-06T16:50:13
736f6a6287c5c74f90a7ce36aa3bbb4b	164753010	2021-10-16T16:16:22	2022-02-02T10:12:50
a61412a9c04683a369264ae4689caba9	175358165	2021-11-26T11:50:58	2021-12-06T09:57:15

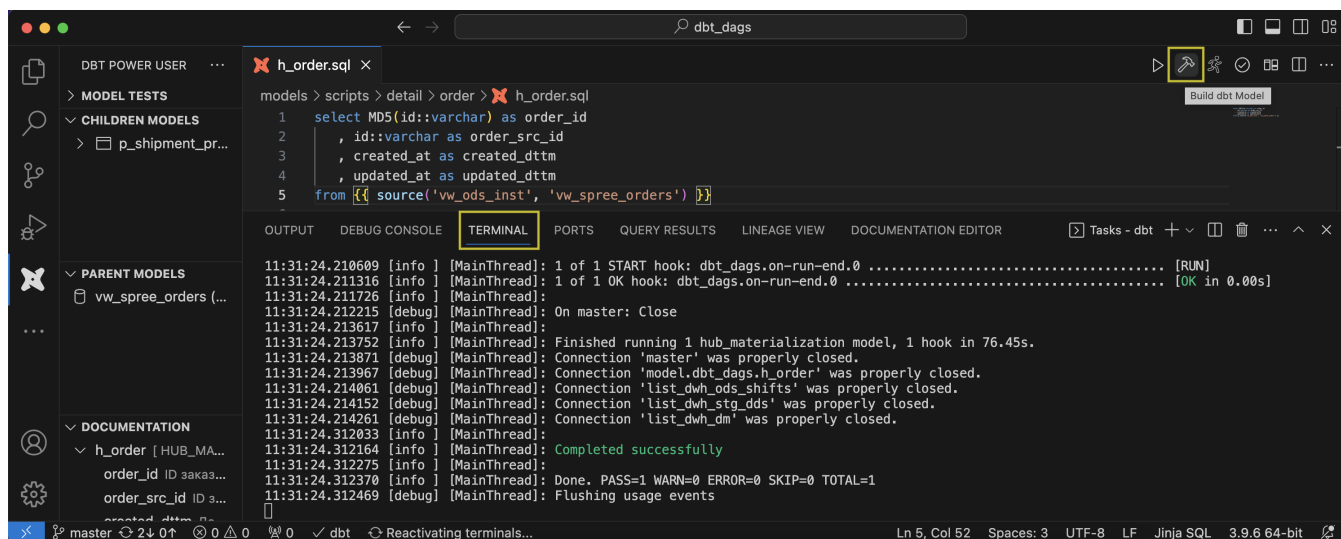
Перейдя в **Dispatched SQL** можно увидеть скомпилированный код (без source и ref конструкций)



## Build dbt Model

При выполнении build сборка модели полностью отработывает и делает create/update конечной таблицы на DEV сервере.

Логи и результаты выполнения команд можно посмотреть на вкладке **TERMINAL**:



## Переход по ссылкам

Навигация к родительской таблице может осуществляться путем зажатия cmd (mac) / ctrl (win) + click на названии таблицы в коде. Вместе с этим можно увидеть описание таблицы и полей, а также их типы:



```
p_shipment_pre_fct_order_tmp.sql X
models > scripts > tmp > p_shipment_pre_fct > p_shipment_pre_fct_order_tmp.sql
40     and axo.mt_active_flg
41     left join {{ ref('s_order_cancellation') }} soc
42         on ord.order_id = soc.order_id
43         and soc.mt_src_cd = 'INST'
44         and not soc.mt_deleted_flg
45         and soc.mt_active_flg
46     left join {{ ref('p_order_marketing_sources') }} oms
47         on ord.order_id = oms.order_id
48 where ord.mt_src_cd = (ref) p_order_marketing_sources
49     and ord.mt_active_flg
50     and not ord.mt_deleted_flg
51
(column) order_id - VARCHAR
(column) order_num - VARCHAR
(column) partner_name - VARCHAR
(column) media_source_name - VARCHAR
(column) traffic_kind_cd - VARCHAR
(column) event_dt - DATE
(column) campaign_name - VARCHAR
(column) campaign_src_id - VARCHAR
(column) retargeting_campaign_flg - BOOLEAN
```

Аналогично можно переместиться к \*.md файлу с описанием, кликнув по нему, находясь в \*.yaml-конфиге.

## Нюансы с TMP

Если в вашем скрипте используются TMP модели, то execute и build выполнить не получится, так как TMP-таблицы создаются при выполнении дага. Работая без Airflow мы лишаем себя такой возможности. Но ЕСЛИ очень надо, можно

1. временно объявить способ материализации TMP-таблиц как CTE. Для этого нужно изменить параметр материализации в файле dbt\_dags/dbt\_project.yml:

```
! dbt_project.yml (Working Tree) 3, M X
! dbt_project.yml > {} models > {} dbt_dags > {} scripts > {} tmp > +materialized
33 dbt_dags:
34   +materialized: table
35   scripts:
36     cte:
37       +materialized: ephemeral
38     tmp:
39       +schema: tmp
40     +materialized: temp_table_materialization
41   vw:
42     +post-hook:
43       sql: "{{ grant_table(this) }}"
44       transaction: true
45
33 dbt_dags:
34   +materialized: table
35   scripts:
36     cte:
37       +materialized: ephemeral
38     tmp:
39       +schema: tmp
40     +materialized: ephemeral
41   vw:
42     +post-hook:
43       sql: "{{ grant_table(this) }}"
44       transaction: true
45
```

2. Убрать из dbt\_project.yml параметр on-run-end и сбилдить необходимые tmp.

```
on-run-end:
- "{{ drop_temp_tables(results) }}"
```

В этом случае они не дропнутся постхуками и будут материализованы в схеме tmp. В конце теста нужно будет удалить все созданные таким образом таблицы из схемы tmp.

Либо можно пойти более простым путем и использовать для теста другую утилиту: [dbt\\_reviewer](#)

## YAML Red Hat

Для более удобной работы с YML можно установить [дополнительное расширение](#), которое позволит видеть структуру конфига на верхней панели:

```
dbt_dags > models > schemas > detail > ! fiscal_receipt.yml > [ ] models > {} 0 > {} config

3 models:
4   - name: h_fiscal_receipt
5     config:
6       materialized: hub_materialization
7       merge_columns:
8         - fiscal_receipt_id
9       process_type: 'detail'
10      src_cd: 'INST'
11      schedule: '40 22 * * *'
12      tags: 'fiscal_receipt'
13      sensor:
14        required_tables:
15          oper:
16            - 'ods_inst.fiscal_receipts'
17      meta:
18        schema: 'dds'
19        distribution: fiscal_receipt_id
20      description: '{{ doc("h_fiscal_receipt") }}'
21      columns:
22        - name: fiscal_receipt_id
```

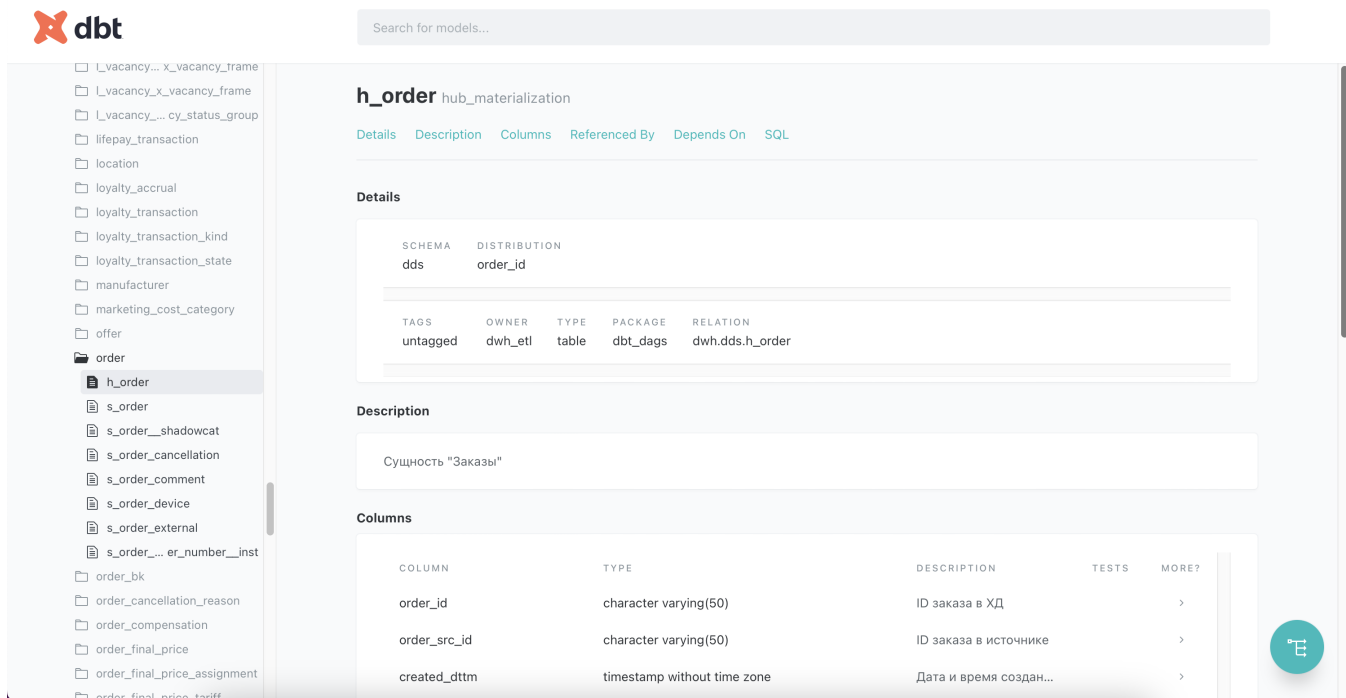
```
# version 2
[ ] models
  { } 0
    name h_fiscal_receipt
  { } config
    description {{ doc("h_fiscal_receipt") }}
  [ ] columns
    { } 1
```

## dbt Docs

Можно поднять локальный dbt Docs с учетом ваших локальных изменений. Для этого нужно выполнить в терминале следующее:

### terminal

```
dbt docs generate; dbt docs serve --port 8001
```



## Полезные ссылки

[Официальный GIT проекта](#)

[Официальный Slack проекта](#)

[Телеграм-чат](#), где на русском языке обсуждается все, что касается dbt