Data Warehouse Analyst

Знакомство с Data Build Tool

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Ставим “+”, если все хорошо
“-”, если есть проблемы

Знакомство с Data Build Tool

Андрей Поляков

В отрасли бэкенд-разработки на Java более 6 лет. Занимался fullstack-разработкой приложений, разработкой высоконагруженных compute-grid систем, а также микросервисов и etl-пайплайнов. Сейчас в роли старшего разработчика работаю над сервисами платежных систем в Unlimint.

Есть опыт работы с сервисами Hadoop (HDFS, HBase), оркестраторами (Airflow, Spring Cloud Data Flow), MPP-базами (Cassandra, Greenplum, Clickhouse).

Интересы: BigData, Blockchain, NFT

Образование: Master Degree in Computer Science and IT, ЮУрГУ, факультет ВШЭКН.

Unlimint

Старший разработчик

Преподает на курсах

- **Highload Architect**
- **Cloud Solution Architecture**
- **Архитектура и шаблоны проектирования**
- **Microservice Architecture**
- **Data Warehouse Analyst**
- **Data Engineer**
- **Java Developer. Basic**

3 года в Otus **261 занятие** **2259 студентов**

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в Telegram @DWH-2024-12

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

Цели вебинара

К концу занятия вы сможете

1. Рассмотреть основные возможности и принципы **dbt**
2. Узнать о конфигурации **dbt**
3. Определить место **dbt** в стеке технологий
4. FAQ по **dbt** и дальнейшее чтение

Data Build Tool (dbt)

Мотивация: SQL в ETL/ELT

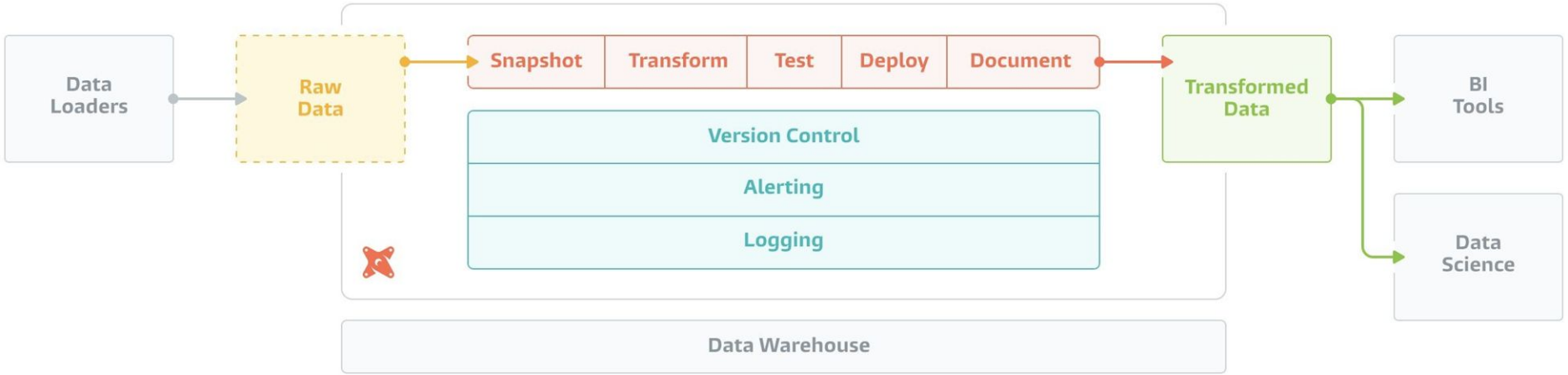
```
CREATE PROCEDURE etl_example AS
BEGIN
-- Extract data from the source table
SELECT * INTO #temp_table FROM source_table;
-- Transform data
UPDATE #temp_table
SET column1 = UPPER(column1),
column2 = column2 * 2;
-- Load data into the target table
INSERT INTO target_table
SELECT * FROM #temp_table;
END
```

Мотивация: ETL-инструменты

```
from airflow import DAG
from airflow.operators.postgres.postgres_operator import PostgresOperator
from datetime import datetime, timedelta

default_args = {
     'owner': 'me',
     'start_date': datetime(2022, 1, 1),
     'depends_on_past': False,
     'retries': 1,
     'retry_delay': timedelta(minutes=5)}
dag = DAG('simple_dag',
     default_args=default_args,
     schedule_interval=timedelta(hours=1))
task1 = PostgresOperator(task_id='get_task_data',
     sql='select * from some_table where 1=1',
     dag=dag)
task2 = PostgresOperator(task_id='upsert_something',
     sql='insert into some_table values()',
     retries=3,
     dag=dag)
task1 >> task2
```

Data Build Tool – T in ELT



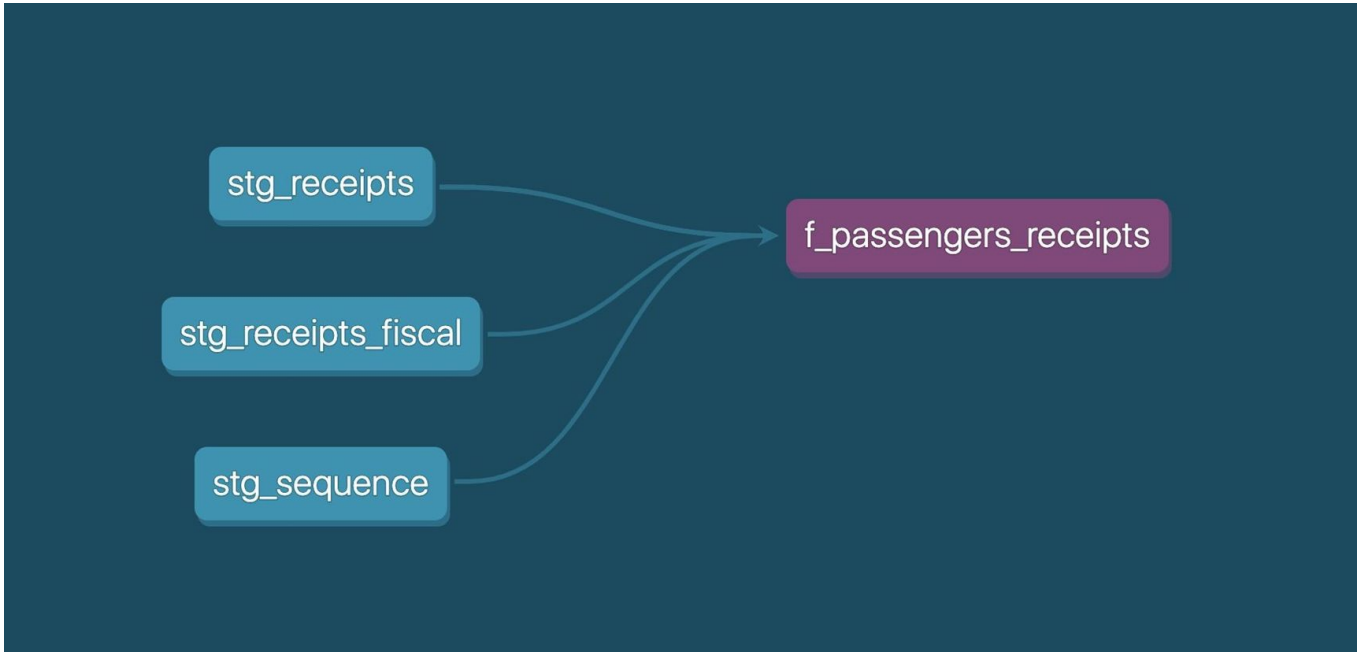
SQL + Jinja

Что делает этот код?

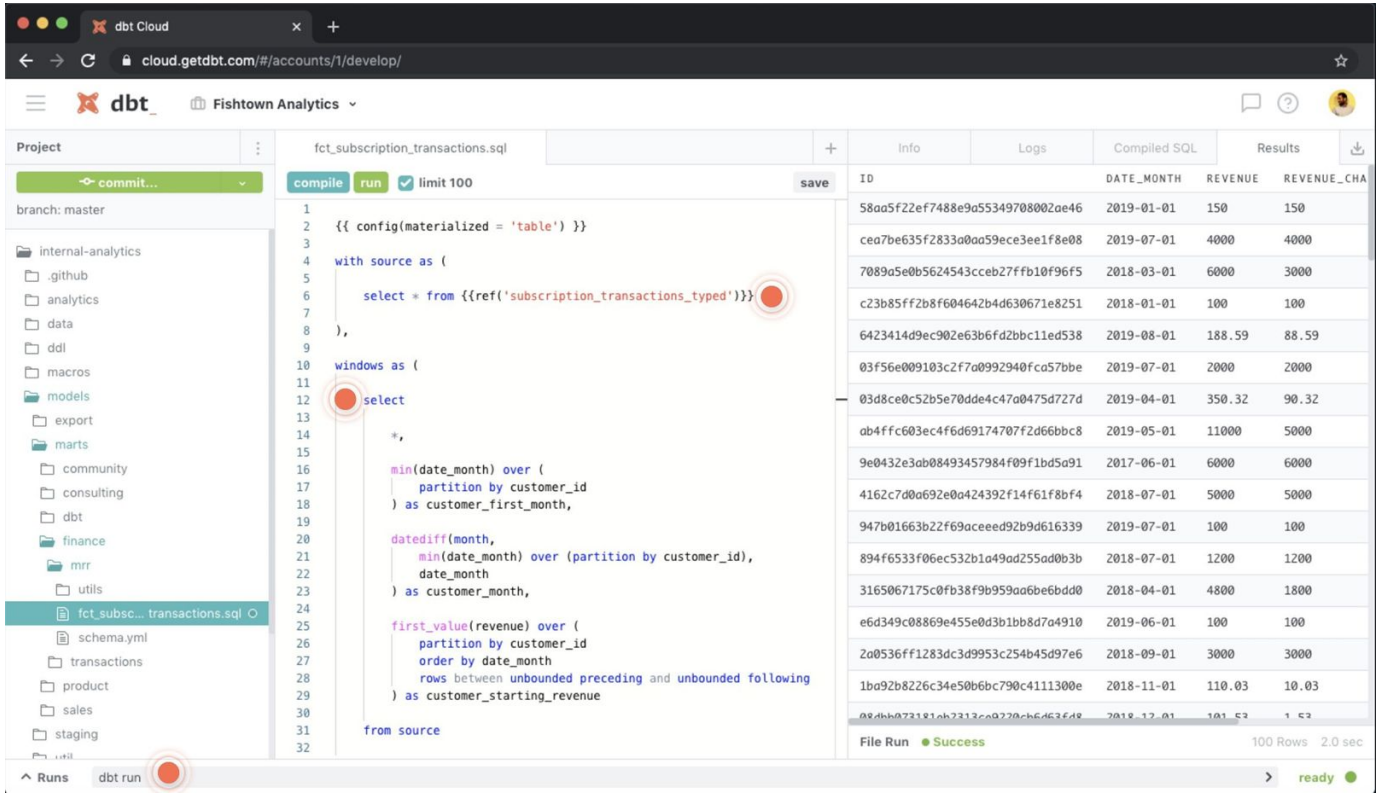
```
select * from event_tracking.events
{% if target.name == ‘dev’ %}
  where created_at >= dateadd(‘day’, -3, current_date)
{% endif %}
```

Сроки выполнения: 1 мин (пишем в чат)

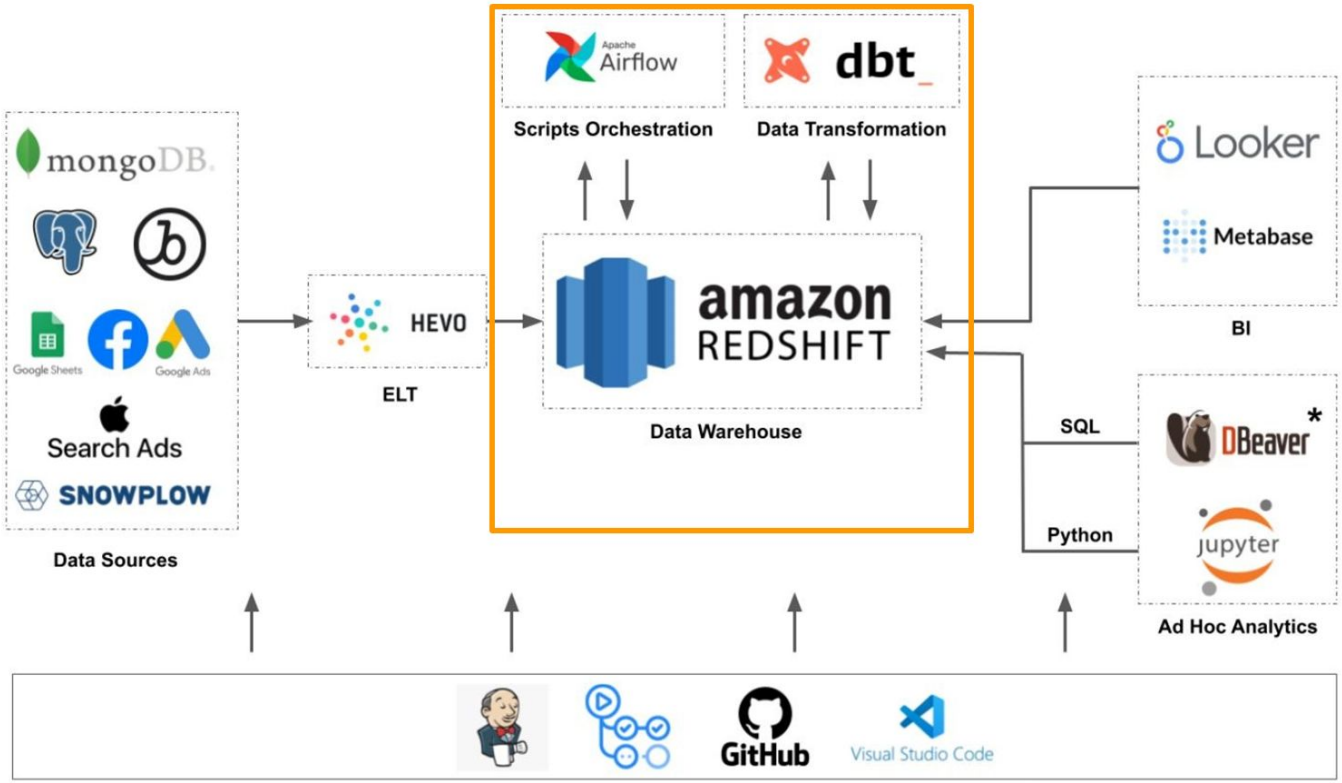
Графы исполнения моделей DAGs



Модели – всё есть SELECT



dbt в Tech Stack @ Wheely



dbt Core vs. dbt Cloud

dbt Core	dbt Cloud
CLI	IDE (Browser based)
Free	Paid $$$
No orchestration (requires Airflow, cron, …)	Built-in orchestration (schedule, triggers, API)
Any adapter you want	Limited number of adapters
-	Additional features: Slim CI, Docs, Source freshness, Slack integration

Конфигурация dbt

dbt project - metadata

```yaml
name: acme corp
profile: acme corp
version: '1.0'

require-dbt-version : ">=0.14.0"

source-paths: ["models"]
analysis-paths: ["analysis"]
test-paths: ["tests"]
data-paths: ["data"]
macro-paths: ["macros"]

target-path: "target"
clean-targets:
    - "target"
    - "dbt_modules"
```

Настройки подключения

```
acme corp:
 outputs:
   dev:
     type: postgres
     threads: 8
     host: [hostname]
     user: [username]
     pass: [password]
     port: 5439
     dbname: [database name]
     schema: dbt_[username]  # e.g. dbt_alice
 target: dev
```

Полная и инкрементальная загрузка

```
{{
    config (
    materialized='incremental',
    sql_where='true',
    unique_key='id',
        dist="call_id",
        sort="min_event_ts_msk",
    )
}}
```

CTE

```
WITH cte_name (column1, column2, ..., columnN) AS (  ❶
    -- Query definition goes here                    ❷
)
SELECT column1, column2, ..., columnN                ❸
FROM cte_name                                        ❸
-- Additional query operations go here               ❹
```

Что такое CTE? Для чего они нужны?

Сроки выполнения: 1 мин (пишем в чат)

CTE

Без CTE

```sql
SELECT pb.book_id,
       pb.title,
       pb.author,
       s.total_sales
FROM (
    SELECT book_id,
           title,
           author
    FROM books
    WHERE rating >= 4.6
) AS pb
JOIN sales s ON pb.book_id = s.book_id
WHERE s.year = 2022
ORDER BY s.total_sales DESC
LIMIT 5;
```

С CTE

```sql
WITH popular_books AS (
    SELECT book_id,
           title,
           author
    FROM books
    WHERE rating >= 4.6
),
best_sellers AS (
    SELECT pb.book_id,
           pb.title,
           pb.author,
           s.total_sales
    FROM popular_books pb
    JOIN sales s ON pb.book_id = s.book_id
    WHERE s.year = 2022
    ORDER BY s.total_sales DESC
    LIMIT 5
)
SELECT *
FROM best_sellers;
```

Модели Stage

```sql
/* This should be file stg_books.sql, and it queries the raw table to create
the new model */

SELECT
  book_id,
  title,
  author,
  publication_year,
  genre
FROM
  raw_books
```

Модели Intermediate

```sql
-- This should be file int_book_authors.sql

-- Reference the staging models
WITH
  books AS (
    SELECT *
    FROM {{ ref('stg_books') }}
  ),
  authors AS (
    SELECT *
    FROM {{ ref('stg_authors') }}
  )

-- Combine the relevant information
SELECT
  b.book_id,
  b.title,
  a.author_id,
  a.author_name
FROM
  books b
JOIN
  authors a ON b.author_id = a.author_id
```

Модели Mart

```sql
-- This should be file mart_book_authors.sql

{{
  config(
    materialized='table',
    unique_key='author_id',
    sort='author_id'
  )
}}

WITH book_counts AS (
  SELECT
    author_id,
    COUNT(*) AS total_books
  FROM {{ ref('int_book_authors') }}
  GROUP BY author_id
)
SELECT
  author_id,
  total_books
FROM book_counts
```

Schema testing

- ✓ Not null
- ✓ Parent-child relationships
- ✓ Expression tests
- ✓ Uniqueness
- ✓ Accepted values
- ✓ Custom data tests

```yaml
version: 2
models:
- name: my_model
  tests:
  - not_null_columns:
    columns:
    - column1
    - column2
```

Документирование

schema.yml

```yaml
version: 2
models:
  - name: events
    description: '{{ doc("table_events") }}'
    columns:
      - name: event_id
        description: This is a unique identifier for the event
        test:
          - unique
          - not_null
```

LIVE DEMO

CLI утилита dbt

Init

Инициализация проекта
https://docs.getdbt.com/reference/commands/init

Пример:
dbt init

Seed

Загружает данные из csv файлов в таблицы БД

Good use-cases for seeds:

- A list of mappings of country codes to country names
- A list of test emails to exclude from analysis
- A list of employee account IDs

Poor use-cases of dbt seeds:

- Loading raw data that has been exported to CSVs
- Any kind of production data containing sensitive information. For example personal identifiable information (PII) and passwords.

Пример:

```
dbt seed
```

run, compile, build

Lifecycle:

- verify
- **parse (verify and create graph)**
- **compile (create sql)**
- **show (compile + run query)**
- **run (compile with change db)**
- **test**
- **build**

Опции:
--select a specific node by name
--inline an arbitrary dbt-SQL query

Tagging models and running subgraphs

```
1. dbt run-operation stage_external_sources --vars 'ext_full_refresh: true'
2. dbt seed
3. dbt run-operation create_udf
4. dbt run --exclude cars_positions_zones tag:dq
5. dbt snapshot
```

```
1. dbt test --schema --exclude f_chauffeurs_sessions_corrected
2. dbt test --data
```

```
1. dbt run -m tag:dq --full-refresh
```

```yaml
wheely:
    +materialized: view
    staging:
        +schema: staging
        +tags: ["staging"]
        braze:
            +schema: braze
            +tags: ["braze"]
    flatten:
        +schema: flatten
        +materialized: incremental
        +unique_key: _id
        +dist: _id
        +sort: _id
        wheely_prod:
            +tags: ["flatten", "wheely_prod"]
        receipt_prod:
            +tags: ["flatten", "receipt_prod"]
    intermediate:
        +schema: intermediate
        +tags: ["intermediate"]
    marts:
        +tags: ["marts"]
        snapshots:
            +tags: ["snapshots"]
        braze:
            +schema: braze
            +materialized: table
            +tags: ["braze"]
```

Node selection syntax

```
# multiple arguments can be provided to --models
$ dbt run --models my_first_model my_second_model

# these arguments can be projects, models, directory paths, tags, or sources
$ dbt run --models tag:nightly my_model finance.base.*

# use methods and intersections for more complex selectors
$ dbt run --models path:marts/finance,tag:nightly,config.materialized:table
```

```
$ dbt run --models my_model+          # select my_model and all children
$ dbt run --models +my_model          # select my_model and all parents
$ dbt run --models +my_model+         # select my_model, and all of its parents and children
```

Extensibility – модульная структура

Importing modules – dbt utilities

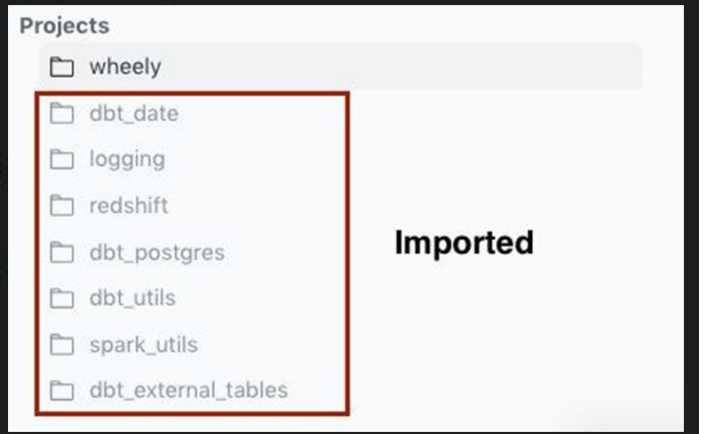
Importing modules allows reusing code

packages.yml

Artemiy Kzr, 2 months ago | 3 authors (Artemiy Kzr and others)

```yaml
packages:
  - package: fishtown-analytics/dbt_utils
    version: 0.6.4
  - package: fishtown-analytics/redshift
    version: 0.4.1
  - package: fishtown-analytics/logging
    version: 0.4.1
  - package: fishtown-analytics/dbt_external_tables
    version: 0.6.2
  - git: "https://github.com/wheely/dbt-date.git"
    revision: 0.2.4
```

Artemiy Kzr, 2 months ago via PR #846 • Ak (#846)



Generating calendar in one line

models > marts > dim > dim_calendar.sql

You, a year ago | 1 author (You)

```
{{
    config(
        materialized='table',
        dist="all",
        sort='date_day'
    )
}}

{{ dbt_date.get_date_dimension('2012-01-01', '2025-12-31') }}
```

	Value
date_day	2021-03-29
prior_date_day	2021-03-28
next_date_day	2021-03-30
prior_year_date_day	2020-03-29
prior_year_over_year_date_day	2020-03-30
day_of_week	1
day_of_week_name	Monday
day_of_week_name_short	Mon
day_of_month	29
day_of_year	88
week_start_date	2021-03-29
week_end_date	2021-04-04
prior_year_week_start_date	2020-03-30
prior_year_week_end_date	2020-04-05
week_of_year	13
iso_week_start_date	2021-03-29
iso_week_end_date	2021-04-04
prior_year_iso_week_start_date	2020-03-30
prior_year_iso_week_end_date	2020-04-05
iso_week_of_year	13
prior_year_week_of_year	14
month_of_year	3
month_name	MARCH
month_name_short	MAR
month_start_date	2021-03-01
month_end_date	2021-03-31
prior_year_month_start_date	2020-03-01
prior_year_month_end_date	2020-03-31
quarter_of_year	1
quarter_start_date	2021-01-01
quarter_end_date	2021-03-31
year_number	2,021
year_start_date	2021-01-01
year_end_date	2021-12-31
fiscal_week_of_year	9

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Рефлексия

Список материалов для изучения

1. dbt Getting Started Tutorial
2. dbt Documentation
3. dbt FAQ
4. How we structure our dbt projects
5. The Modern Data Stack: Past, Present, and Future
6. Five principles that will keep your data warehouse organized
7. The Analytics Engineering Guide

Делитесь своими материалами в Slack

Заполните, пожалуйста, опрос о занятии по ссылке в чате

Спасибо за внимание!