

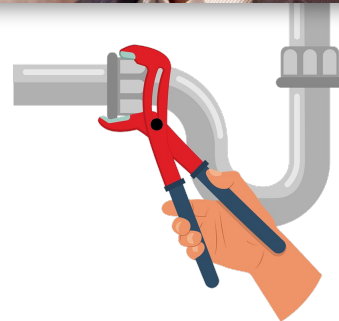


dbt
_

Why it's awesome and how we use it in MoneyLion

WHO AM I

- Boon Keong (BK)
- Work for MoneyLion (US fintech company)
- A data engineer
- Build and maintain data pipeline architectures that enables analytic teams to work more efficiently
- Even if things gets messy 💩



MOTIVATION

WHAT IS DBT

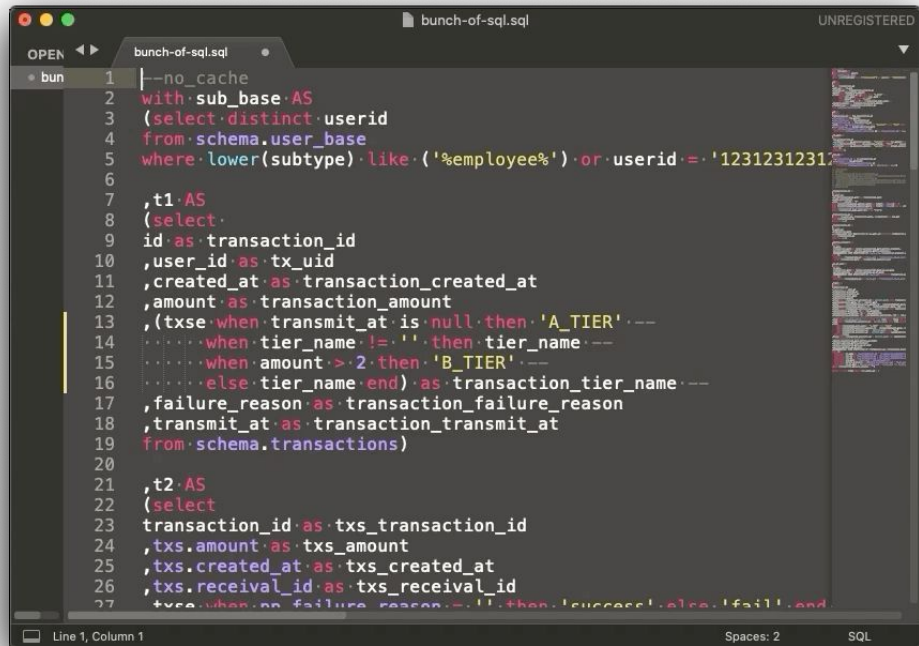
PROBLEM-SOLUTION

PRODUCTIONISATION

ANALYTICS TRANSFORMED

TYPICAL ANALYTICS

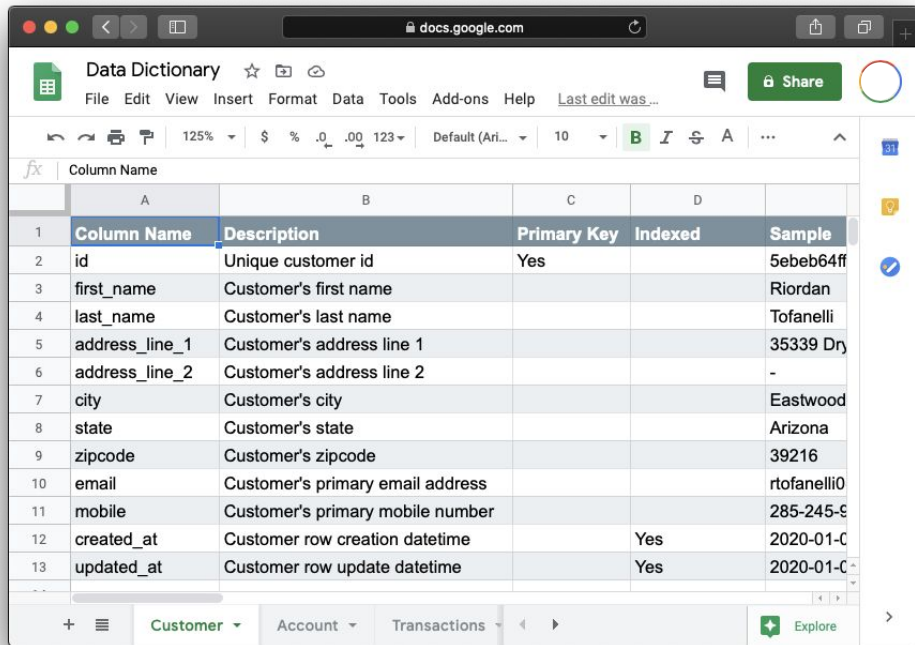
- Messy (copy paste) SQL code



```
1  |--no_cache
2  with sub_base AS
3  (select distinct userid
4  from schema.user_base
5  where lower(subtype).like ('%employee%') or userid = '1231231231'
6
7  ,t1 AS
8  (select
9  id as transaction_id
10 ,user_id as tx_uid
11 ,created_at as transaction_created_at
12 ,amount as transaction_amount
13 ,(txse.when.transmit_at is null then 'A_TIER' ---
14  ....when tier_name != '' then tier_name ---
15  ....when amount > 2 then 'B_TIER' ---
16  ....else tier_name end) as transaction_tier_name ---
17 ,failure_reason as transaction_failure_reason
18 ,transmit_at as transaction_transmit_at
19 from schema.transactions)
20
21 ,t2 AS
22 (select
23 transaction_id as txs_transaction_id
24 ,txs.amount as txs_amount
25 ,txs.created_at as txs_created_at
26 ,txs.receive_id as txs_receive_id
27 ,txs.when.no_failure_reason = '' then 'success' else 'fail' end
```

TYPICAL ANALYTICS

- Messy (copy paste) SQL code
- Little or no documentation



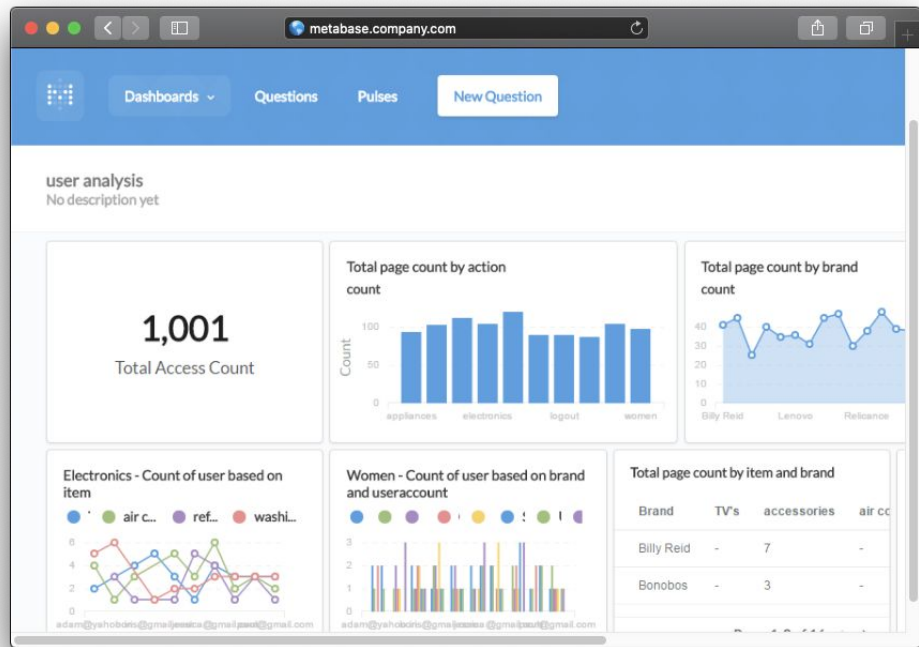
The screenshot shows a Google Docs spreadsheet titled "Data Dictionary". The spreadsheet contains a table with the following columns: Column Name, Description, Primary Key, Indexed, and Sample. The table lists various database columns for a customer database, including id, first_name, last_name, address_line_1, address_line_2, city, state, zipcode, email, mobile, created_at, and updated_at.

	A	B	C	D	
	Column Name	Description	Primary Key	Indexed	Sample
1	id	Unique customer id	Yes		5ebeb64ff
2	first_name	Customer's first name			Riordan
3	last_name	Customer's last name			Tofanelli
4	address_line_1	Customer's address line 1			35339 Dry
5	address_line_2	Customer's address line 2			-
6	city	Customer's city			Eastwood
7	state	Customer's state			Arizona
8	zipcode	Customer's zipcode			39216
9	email	Customer's primary email address			rtofanelli0
10	mobile	Customer's primary mobile number			285-245-9
11	created_at	Customer row creation datetime		Yes	2020-01-C
12	updated_at	Customer row update datetime		Yes	2020-01-C

At the bottom of the spreadsheet, there are tabs for "Customer", "Account", and "Transactions". A green "Explore" button is visible in the bottom right corner.

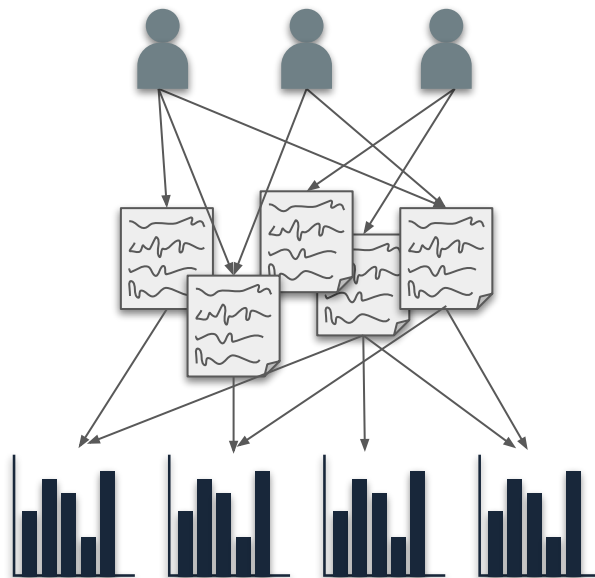
TYPICAL ANALYTICS

- Messy (copy paste) SQL code
- Little or no documentation
- But simple BI requirements



PAIN POINTS

- Disorganised & difficult to collaborate



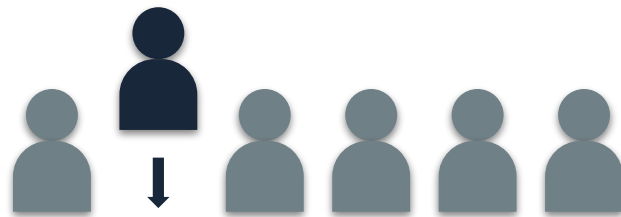
PAIN POINTS

- Disorganised & difficult to collaborate
- Hard to visualise links between tables



PAIN POINTS

- Disorganised & difficult to collaborate
- Hard to visualise links between tables
- Analysis on-boarding / handover is a pain



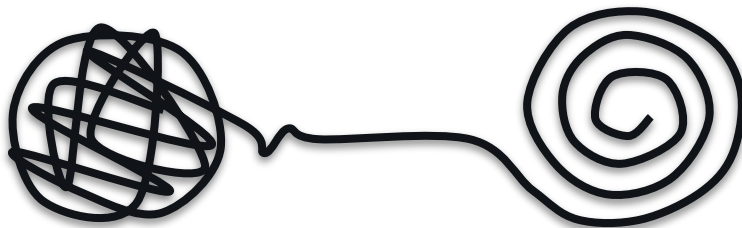
WE WANT

- A way to organised code base for collaboration



WE WANT

- A way to organised code base for collaboration
- A way to break down complex SQL code for better comprehension



WE WANT

- A way to organised code base for collaboration
- A way to break down complex SQL code for better comprehension
- A simple and easy way to document tables for handovers or reference



WE WANT

- A way to organised code base for collaboration
- A way to break down complex SQL code for better comprehension
- A simple and easy way to document tables for handovers or reference



MOTIVATION

WHAT IS DBT

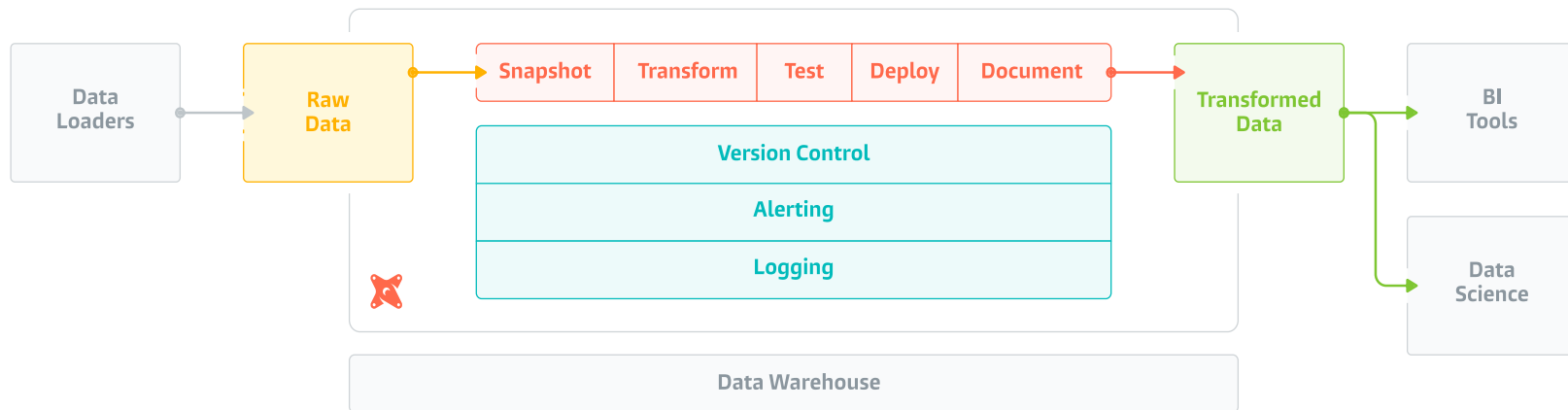
PROBLEM-SOLUTION

PRODUCTIONISATION

ANALYTICS TRANSFORMED

WHAT IS DBT (data build tool)

- “a toolkit for building analytics the way developers build applications”
—data engineering podcast, episode 81
- Handles the “**T**” in “EL**T**” (Extract-Load-**Transform**)



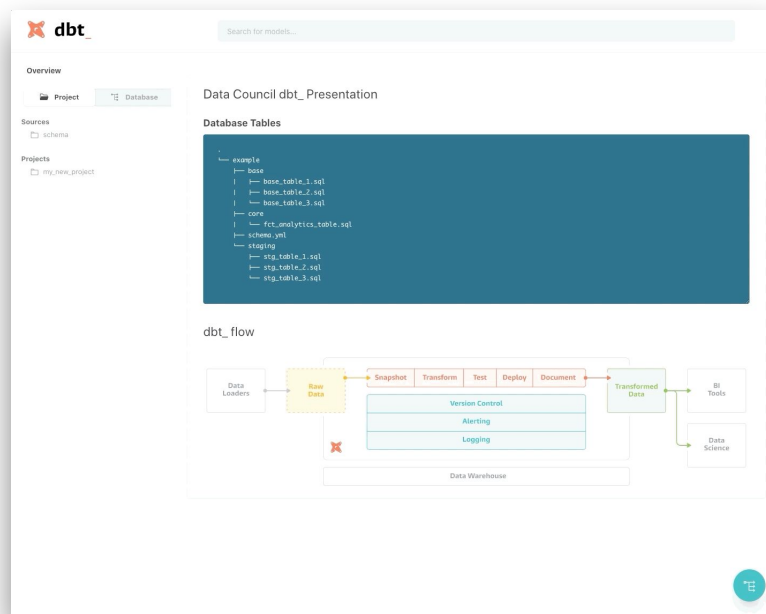
WHY IT'S AWESOME BEFORE

```

1 WITH employee AS
2 (
3   --SELECT DISTINCT id
4   --FROM (SELECT id
5   --FROM schema.subs
6   --WHERE LOWER(subtype) LIKE ('%employee%')
7   --OR id = '123123123123'
8   --UNION
9   --SELECT id
10  --FROM schema.users
11  --WHERE email LIKE '%@company.com')
12 ),
13 req_final_cte AS
14 (
15   --SELECT id ..... AS req_id,
16   --user_id ..... AS user_id,
17   --created_at ..... AS laon_reqld_at,
18   --effective_date ..... AS laon_effective_at,
19   --disbursed_at ..... AS laon_disbursed_at,
20   --"type" ..... AS release_type,
21   --status ..... AS high_level_status,
22   --failure_reason ..... AS req_failure_reason,
23   --receival_at ..... AS laon_receival_at,
24   --txsE
25   --WHEN laon_receival_at < CURRENT_DATE THEN TRUE
26   --ELSE FALSE
27   --END AS laon_due,
28   --original_maximum_eligible AS original_maximum_eligible,
29   --max_amount ..... AS laon_max_amount,
30   --min_amount ..... AS laon_min_amount,
31   --available_amount ..... AS laon_available_amount,
32   --txsE
33   --WHEN receival_at IS NULL AND laon_amount = 2 THEN 'A_TIER'
34   --WHEN tier_name = '' AND laon_amount > 2 THEN 'B_TIER'
35   --ELSE tier_name
36   --END AS laon_tier_name,
37   --laon_amount ..... AS laon_amount_on_req
38   --FROM schema.tx_req
39 ),
40 tx_cdd_cte AS
41 (
42   --SELECT tx.id ..... AS laon_id,

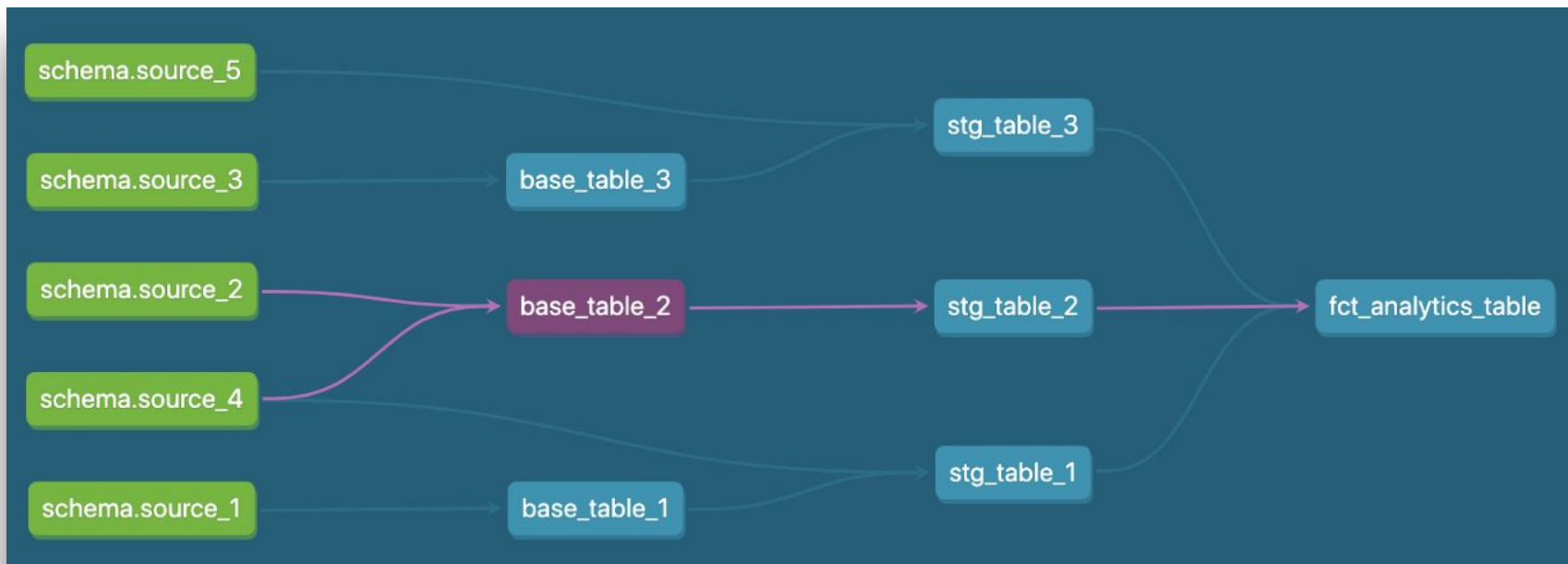
```

AFTER



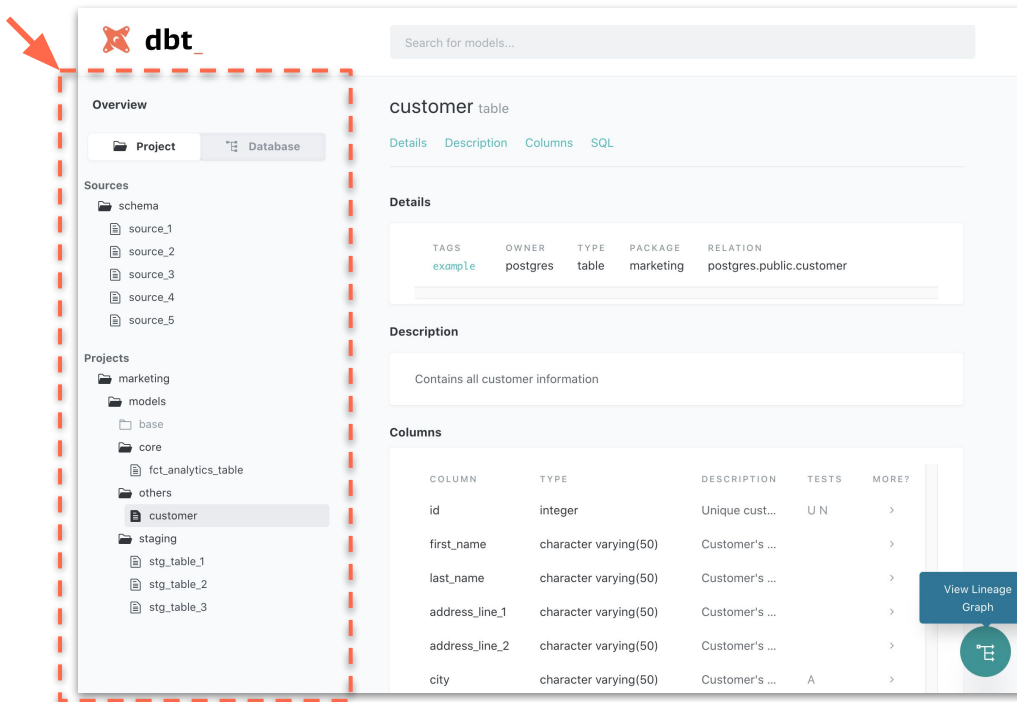
WHY IT'S AWESOME

DATA LINEAGE VISUALISATION



WHY IT'S AWESOME

DATA MODELS



The screenshot displays the dbt_ interface for a data model named 'customer'. The left sidebar, enclosed in a red dashed box, shows the project structure with 'customer' selected under the 'models' folder. The main content area provides details for the 'customer' table, including its description, columns, and a 'View Lineage Graph' button.

Overview

Project Database

Sources

- schema
 - source_1
 - source_2
 - source_3
 - source_4
 - source_5

Projects

- marketing
- models
 - base
 - core
 - fct_analytics_table
 - others
 - customer**
 - staging
 - stg_table_1
 - stg_table_2
 - stg_table_3

customer table

Details Description Columns SQL

Details

TAGS	OWNER	TYPE	PACKAGE	RELATION
example	postgres	table	marketing	postgres.public.customer

Description

Contains all customer information

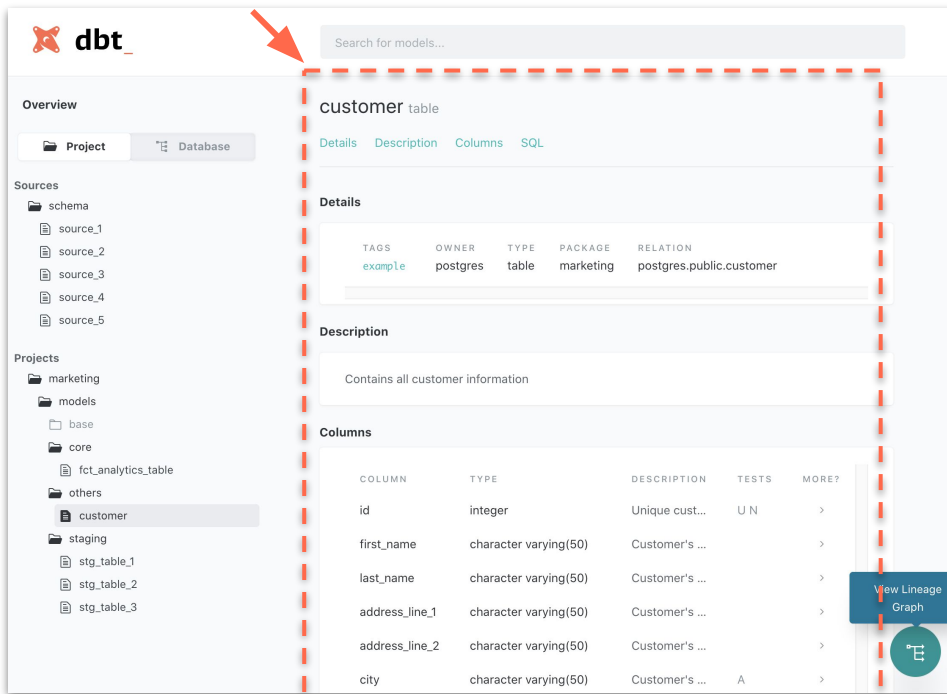
Columns

COLUMN	TYPE	DESCRIPTION	TESTS	MORE?
id	integer	Unique cust...	U N	>
first_name	character varying(50)	Customer's ...		>
last_name	character varying(50)	Customer's ...		>
address_line_1	character varying(50)	Customer's ...		>
address_line_2	character varying(50)	Customer's ...		>
city	character varying(50)	Customer's ...	A	>

[View Lineage Graph](#)

WHY IT'S AWESOME

DATA DICTIONARY



The screenshot displays the dbt Data Dictionary interface. A red arrow points to the search bar at the top. The left sidebar shows a tree view of sources and projects, with the 'customer' model selected under the 'marketing' project. The main content area shows the details for the 'customer' table, including its description and a list of columns.

Overview

Search for models...

customer table

[Details](#) [Description](#) [Columns](#) [SQL](#)

Details

TAGS	OWNER	TYPE	PACKAGE	RELATION
example	postgres	table	marketing	postgres.public.customer

Description

Contains all customer information

Columns

COLUMN	TYPE	DESCRIPTION	TESTS	MORE?
id	integer	Unique cust...	UN	>
first_name	character varying(50)	Customer's ...		>
last_name	character varying(50)	Customer's ...		>
address_line_1	character varying(50)	Customer's ...		>
address_line_2	character varying(50)	Customer's ...		>
city	character varying(50)	Customer's ...	A	>

[View Lineage Graph](#)

MOTIVATION

WHAT IS DBT

PROBLEM-SOLUTION

PRODUCTIONISATION


ANALYTICS TRANSFORMED

#1 - GET ORGANIZED

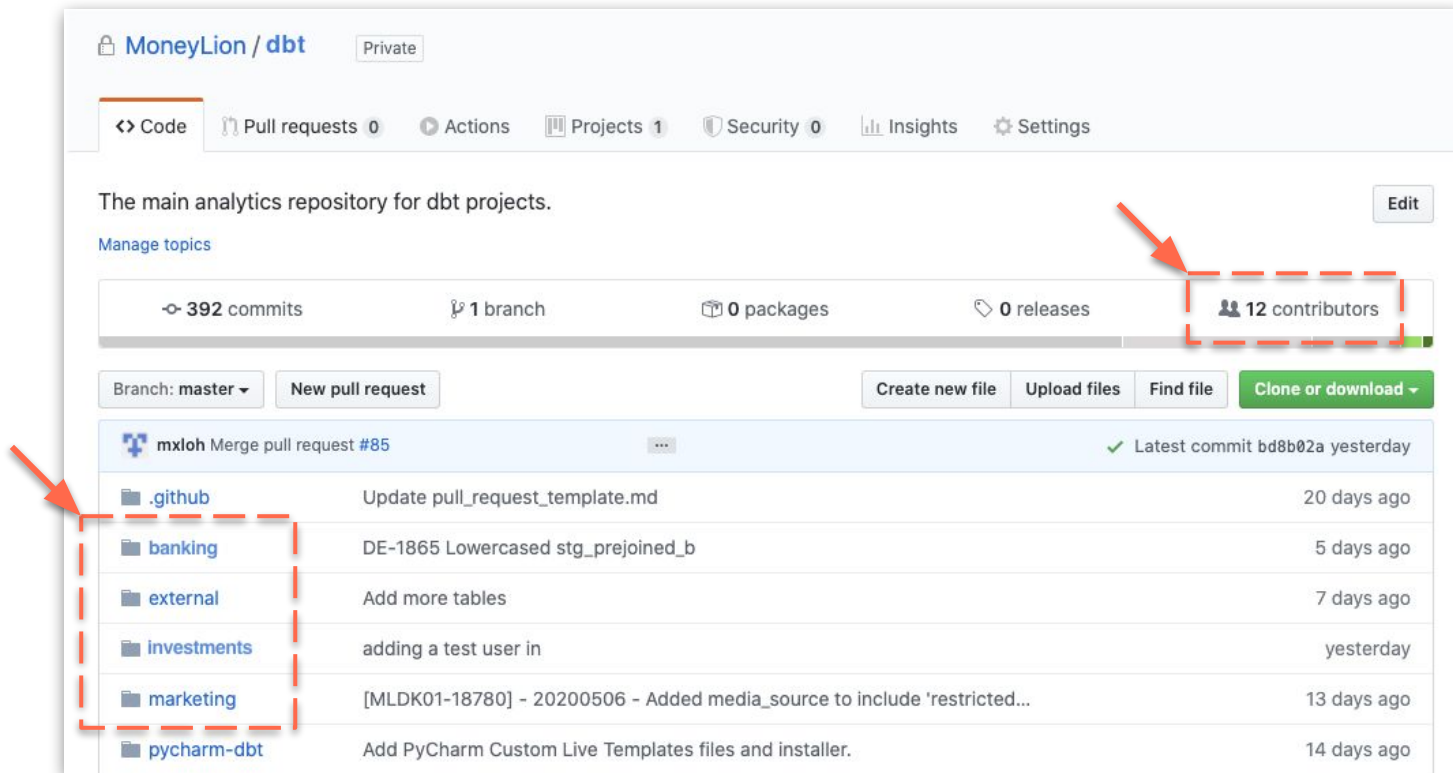
PROBLEM

- Disorganised SQL code
 - Stored in text files ✗
 - Send via email/messengers ✗
 - Difficult to collaborate ✗

SOLUTION

- Use dbt with  GitHub
 - Transparency ✓
 - Version controlled ✓
 - Easier collaboration ✓

GET ORGANIZED



MoneyLion / dbt Private

<> Code Pull requests 0 Actions Projects 1 Security 0 Insights Settings

The main analytics repository for dbt projects. [Manage topics](#) [Edit](#)

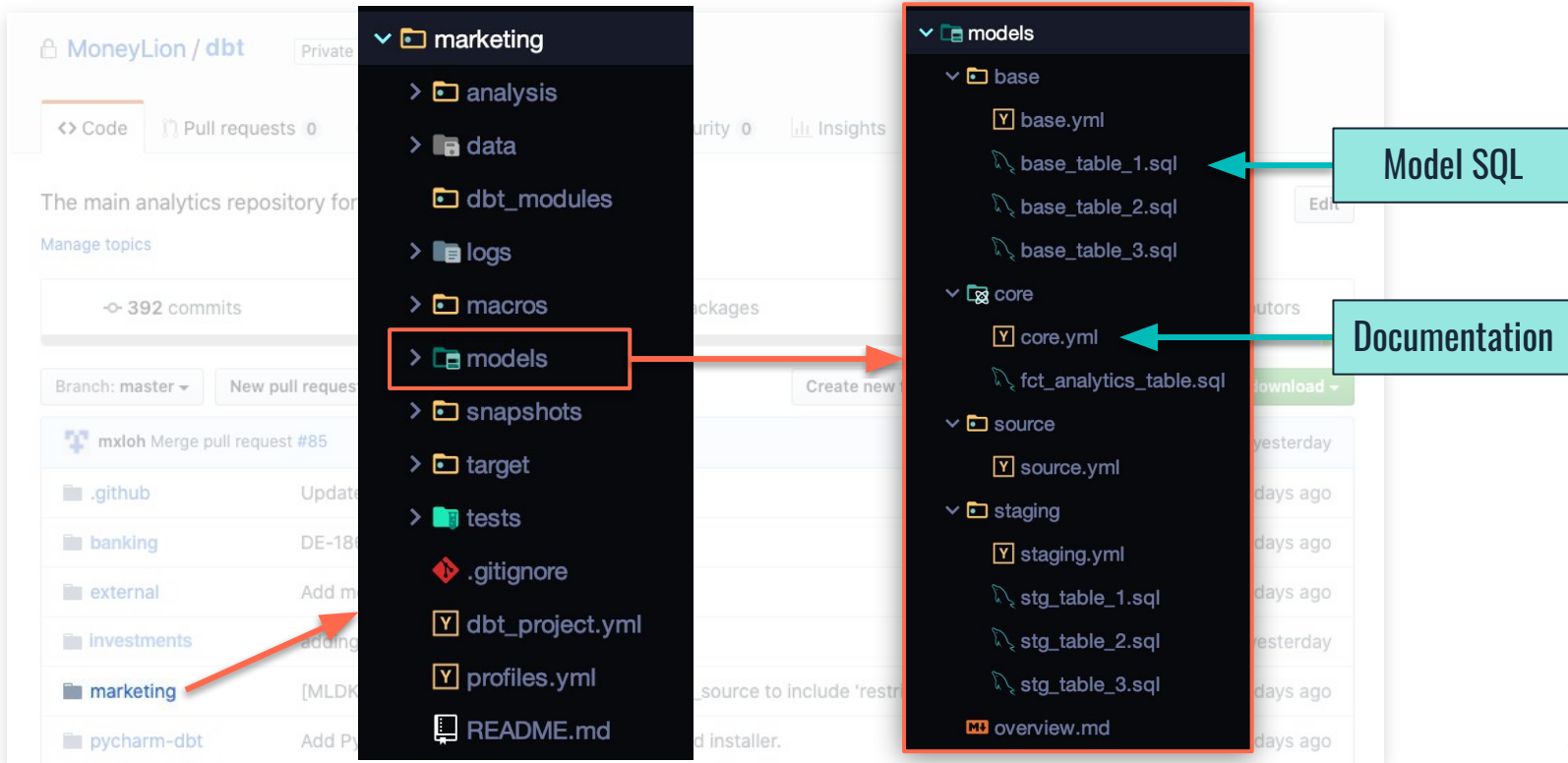
392 commits 1 branch 0 packages 0 releases 12 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

mxloh Merge pull request #85 Latest commit bd8b02a yesterday

.github	Update pull_request_template.md	20 days ago
banking	DE-1865 Lowercased stg_prejoined_b	5 days ago
external	Add more tables	7 days ago
investments	adding a test user in	yesterday
marketing	[MLDK01-18780] - 20200506 - Added media_source to include 'restricted...	13 days ago
pycharm-dbt	Add PyCharm Custom Live Templates files and installer.	14 days ago

GET ORGANIZED



The screenshot displays the GitHub repository structure for MoneyLion/dbt. The left sidebar shows the repository's file tree, with the **marketing** folder highlighted. The main content area shows the contents of the **models** folder, which is also highlighted. A red arrow points from the **marketing** folder in the sidebar to the **models** folder in the main content area. Two blue arrows point from text boxes to specific files: one from **Model SQL** to **base_table_1.sql** and another from **Documentation** to **core.yml**.

marketing

- analysis
- data
- dbt_modules
- logs
- macros
- models**
- snapshots
- target
- tests
- .gitignore
- dbt_project.yml
- profiles.yml
- README.md

models

- base
 - base.yml
 - base_table_1.sql
 - base_table_2.sql
 - base_table_3.sql
- core
 - core.yml
 - fct_analytics_table.sql
- source
 - source.yml
- staging
 - staging.yml
 - stg_table_1.sql
 - stg_table_2.sql
 - stg_table_3.sql
- overview.md

#2 - BREAKING IT DOWN

PROBLEM

- SQL too long and complex
 - Hard to understand/visualise ✗
 - Slow/Hard to debug ✗
 - Many repeated CTEs/logic ✗

SOLUTION

- Use **Data Models** concept
 - Reduced cognitive load ✓
 - Easy to debug ✓
 - Reusable code ✓
- Use **Macros** to keep DRY ✓

CTE = Common Table Expression

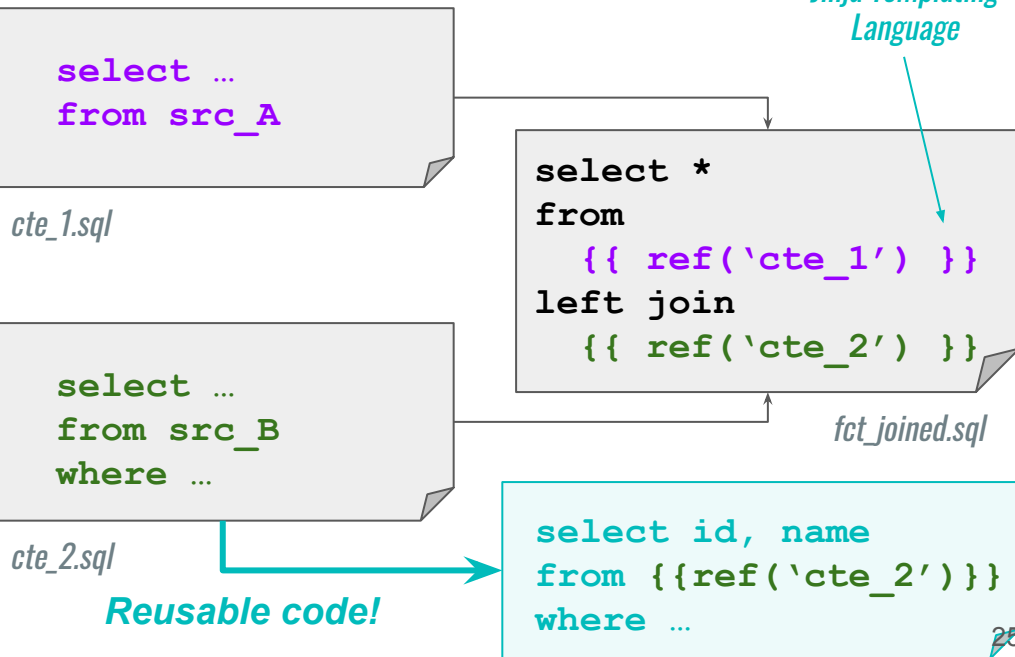
DRY = Don't Repeat Yourself

DATA MODELS

BEFORE

```
with cte_1 as (  
    select ...  
    from src_A  
) ,  
  
cte_2 as (  
    select ...  
    from src_B  
    where ...  
)  
  
select * from cte_1  
left join cte_2
```

AFTER



DATA MODELS

SOURCE

-

BASE

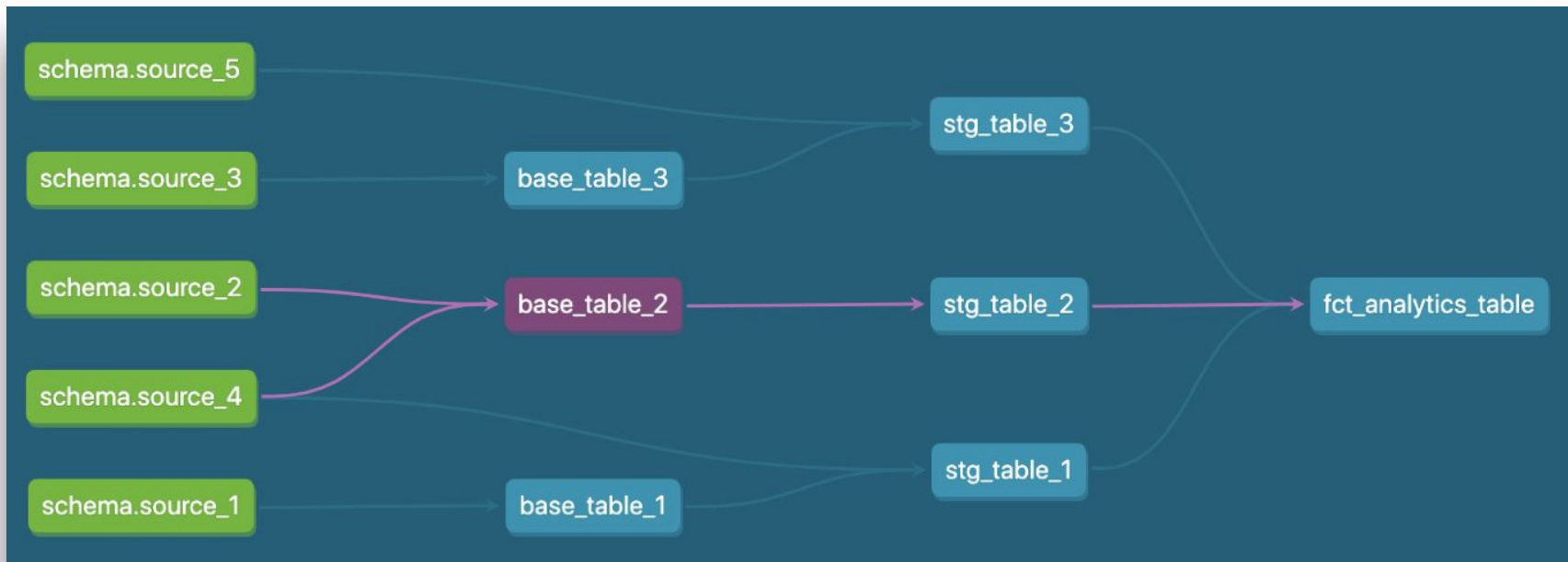
ephemeral

STAGING

view (no-binding)

FACT

table



MACROS

BEFORE

```
select
  case
    when A then 1
    when B then 2
    when C then 3
    when D then 4
    else null
  end as position,
  ... ..
from alphabet
group by 1, 2, 3
```

AFTER

Jinja Templating Language

```
{% macro pos(col) %}
  case
    when A then 1
    when B then 2
    when C then 3
    when D then 4
    else null
  end as position
{% endmacro %}
```

macros.sql

Jinja Templating Language

```
select
  {{ pos(alphabet) }},
  ... ..
from alphabet
{{ group_by(3) }}
```

stg_alphabet.sql

Reusable code!

```
select
  {{ pos(alphabet) }},
  ... ..
from another_table
```

MACROS

BEFORE

```
select
  case
    when alphabet_1 = A then 1
    when alphabet_1 = B then 2
    when alphabet_1 = C then 3
    when alphabet_1 = D then 4
    else null
  end as position_1,
  case
    when alphabet_2 = A then 1
    when alphabet_2 = B then 2
    when alphabet_2 = C then 3
    when alphabet_2 = D then 4
    else null
  end as position_2,
  case
    when alphabet_3 = A then 1
    when alphabet_3 = B then 2
    when alphabet_3 = C then 3
    when alphabet_3 = D then 4
    else null
  end as position_3,
  ... ..
from alphabet
group by 1, 2, 3
```

AFTER

```
select
  {{ pos(alphabet_1) }} as position_1,
  {{ pos(alphabet_2) }} as position_2,
  {{ pos(alphabet_3) }} as position_3,
  ... ..
from alphabet
{{ dbt_utils.group_by(3) }}
```

#3 - DOCUMENT

PROBLEM

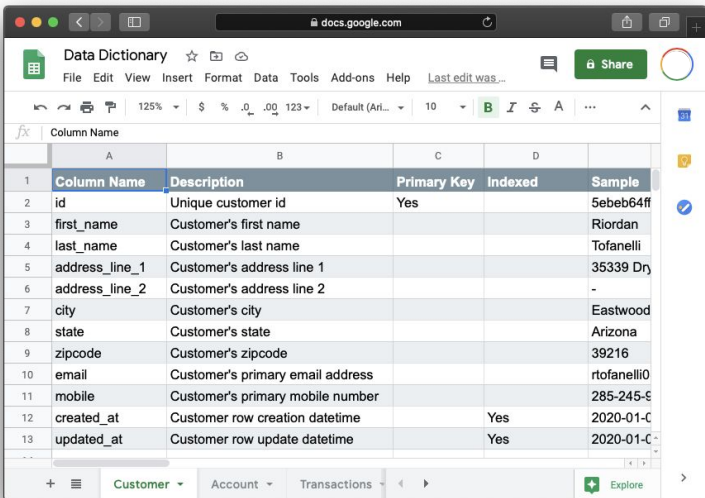
- Lack of Documentation
 - On-boarding/handover pains ✖ ✖
 - Management questions ✖
 - Headache when revisiting ✖

SOLUTION

- Use dbt docs
 - Nice & Sleek UI ✔
 - Visible Lineage ✔
 - Unit Testing & Integrity Checks ✔

DOCUMENTATION

BEFORE



Column Name	Description	Primary Key	Indexed	Sample
id	Unique customer id	Yes		5ebeb64ff
first_name	Customer's first name			Riordan
last_name	Customer's last name			Tofanelli
address_line_1	Customer's address line 1			35339 Dry
address_line_2	Customer's address line 2			-
city	Customer's city			Eastwood
state	Customer's state			Arizona
zipcode	Customer's zipcode			39216
email	Customer's primary email address			rtofanelli0
mobile	Customer's primary mobile number			285-245-9
created_at	Customer row creation datetime		Yes	2020-01-0
updated_at	Customer row update datetime		Yes	2020-01-0

AFTER (in YAML)

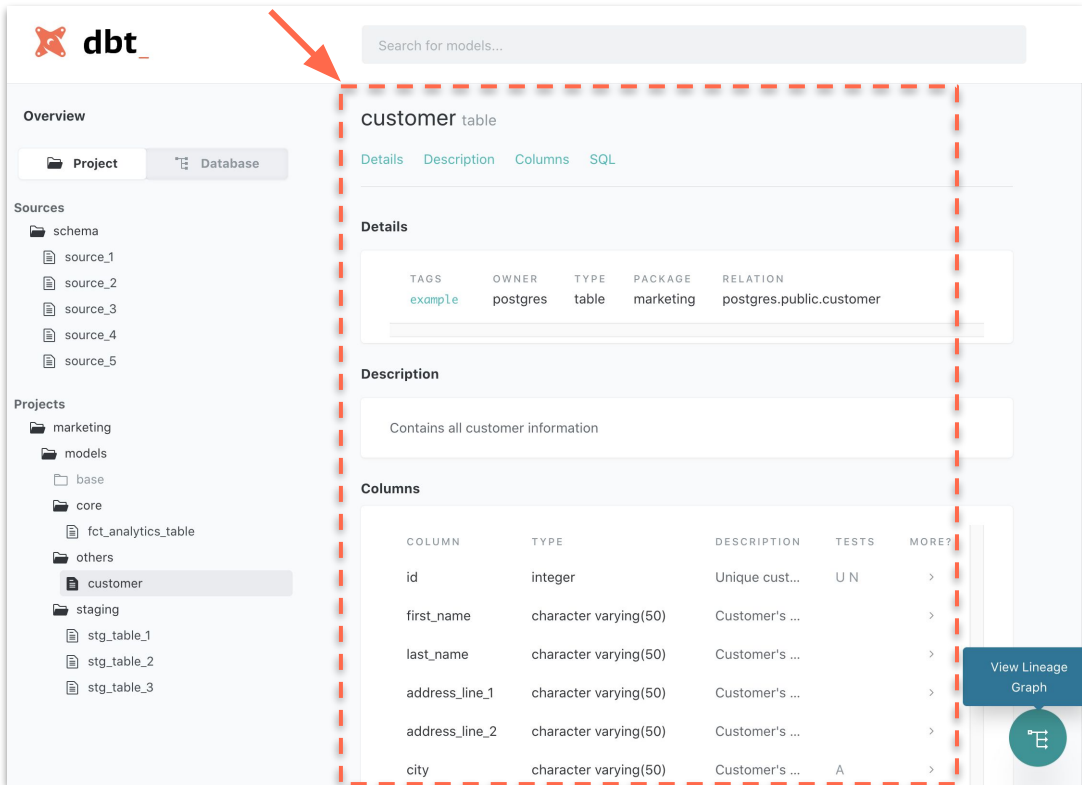
```
models:
  - name: customer
    description: "Contains all customer information"

    columns:
      - name: id
        description: "Unique customer id"
        tests: <2 items>

      - name: first_name
        description: "Customer's first name"

      - name: last_name
        description: "Customer's last name"
```

DATA DICTIONARY



The screenshot displays the dbt Data Dictionary interface. On the left, a sidebar shows the project structure under 'marketing' > 'models' > 'core' > 'customer'. The main panel shows details for the 'customer' table, including its description and a list of columns. A red dashed box highlights the main content area, and a red arrow points to the dbt logo.

dbt

Search for models...

Overview

Project Database

Sources

- schema
 - source_1
 - source_2
 - source_3
 - source_4
 - source_5

Projects

- marketing
 - models
 - base
 - core
 - fct_analytics_table
 - others
 - customer**
 - staging
 - stg_table_1
 - stg_table_2
 - stg_table_3

customer table

Details Description Columns SQL

Details

TAGS	OWNER	TYPE	PACKAGE	RELATION
example	postgres	table	marketing	postgres.public.customer

Description

Contains all customer information

Columns

COLUMN	TYPE	DESCRIPTION	TESTS	MORE?
id	integer	Unique cust...	U N	>
first_name	character varying(50)	Customer's ...		>
last_name	character varying(50)	Customer's ...		>
address_line_1	character varying(50)	Customer's ...		>
address_line_2	character varying(50)	Customer's ...		>
city	character varying(50)	Customer's ...	A	>

View Lineage Graph

DATA TESTING & CHECKS

DEFINE TESTS (in YAML)

```
columns:
  - name: id
    description: "Unique customer id"
    tests:
      - unique
      - not_null
    ...
  - name: city
    description: "Customer's city"
    tests:
      - accepted_values:
          values:
            - 'Kuala Lumpur'
            - 'Kuching'
            - 'Johor Bahru'
            - 'Melaka'
            - 'Ipoh'
```

RUN TESTS

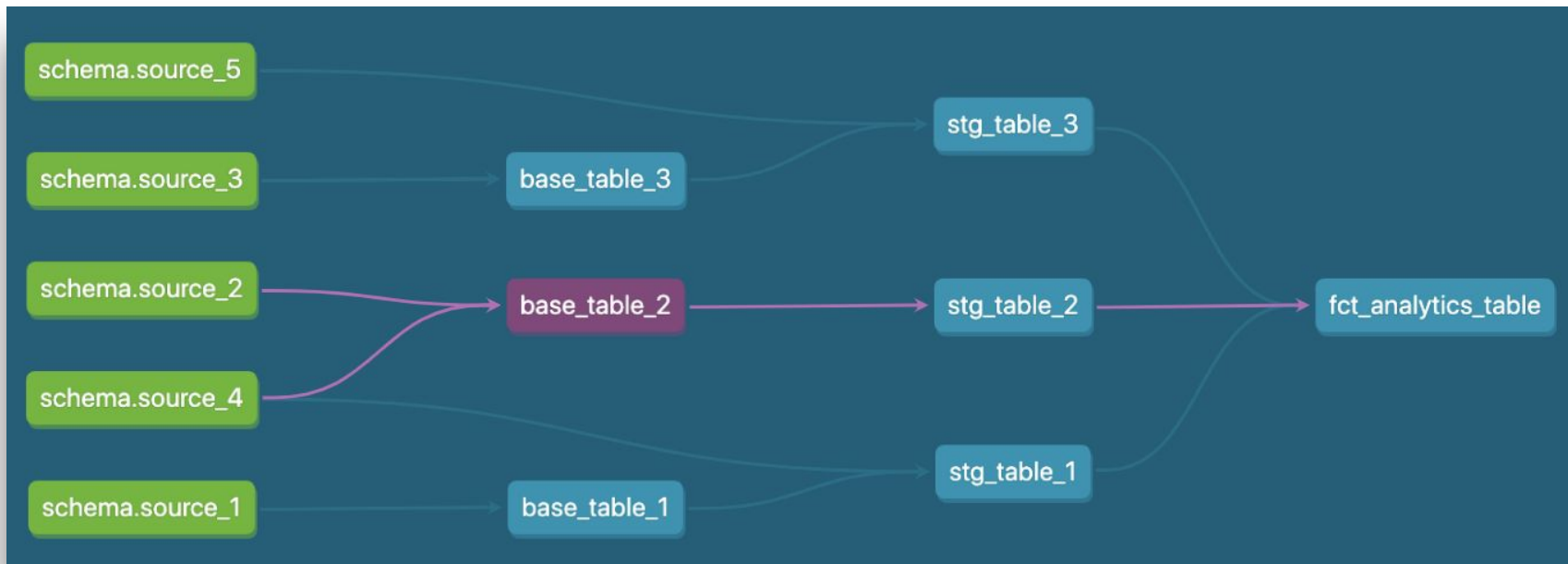
```
bash
$ dbt test
Concurrency: 8 threads (target='dev')

1 of 3 START test accepted_values_customer_city_Kuala_Lumpur_Kuch... [RUN]
2 of 3 START test not_null_customer_id..... [RUN]
3 of 3 START test unique_customer_id..... [RUN]
1 of 3 PASS accepted_values_customer_city_Kuala_Lumpur_Kuch... [PASS in 0.09s]
2 of 3 PASS not_null_customer_id..... [PASS in 0.09s]
3 of 3 PASS unique_customer_id..... [PASS in 0.09s]

Finished running 3 tests in 0.77s.

Completed successfully
```


DATA LINEAGE VISUALISATION



MOTIVATION

WHAT IS DBT

PROBLEM-SOLUTION

PRODUCTIONISATION

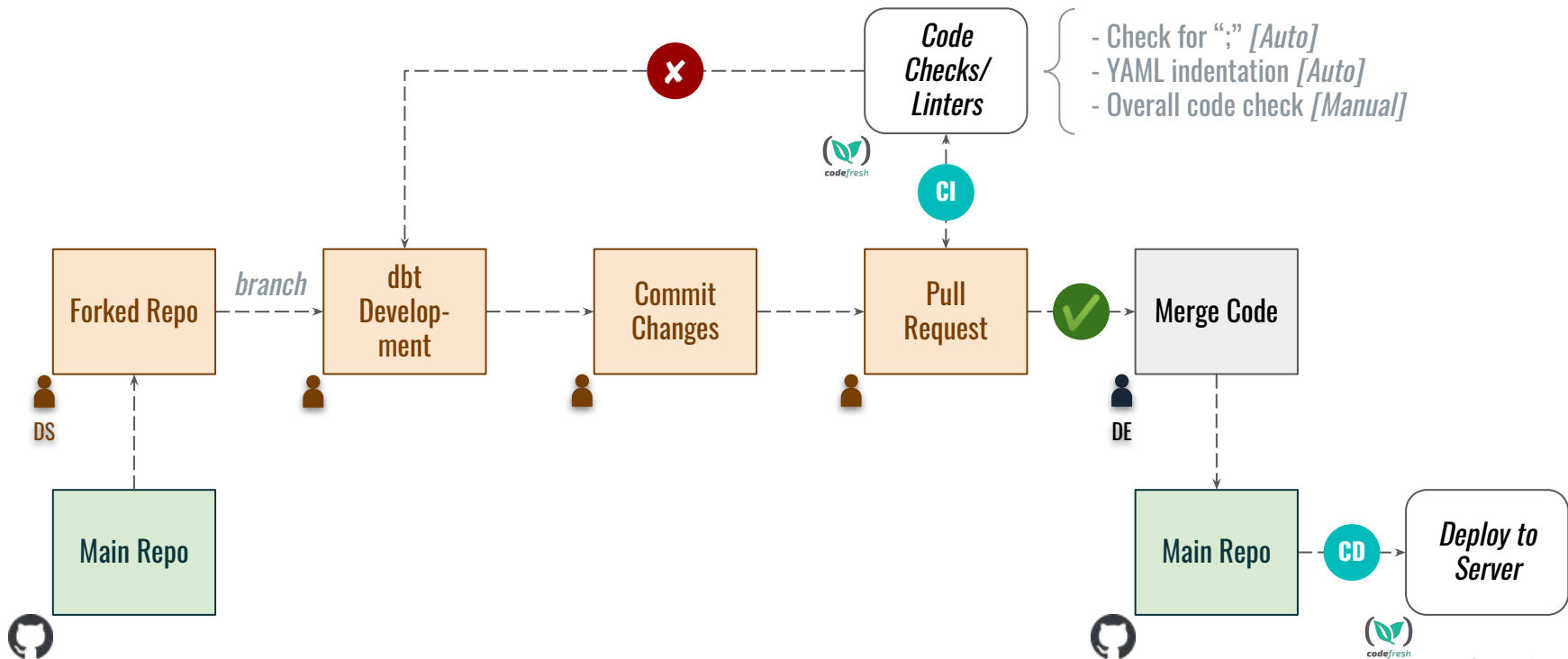
ANALYTICS TRANSFORMED

PRODUCTIONISATION

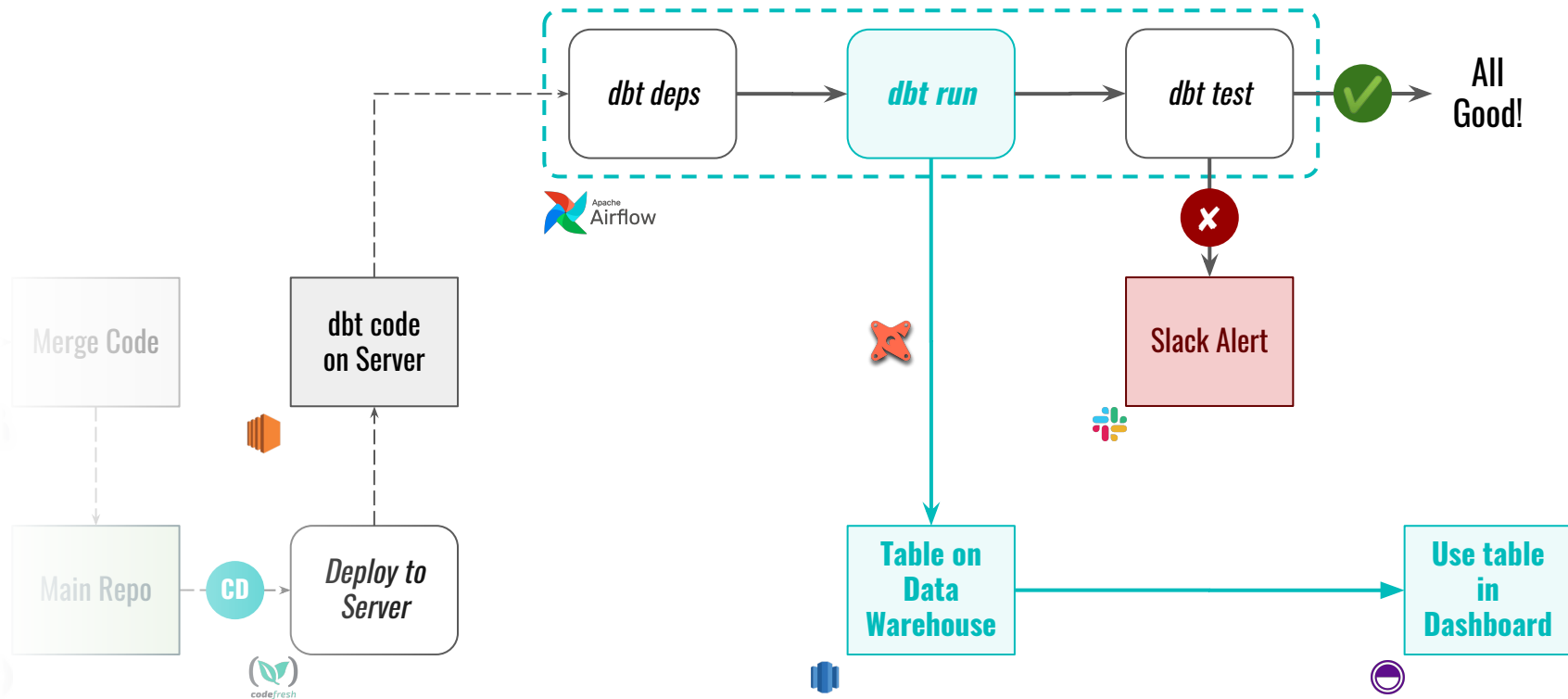
ASK

- How to get people to *contribute*?
- How do we *automate the “T”* pipeline?
- Where do I put the *documentation* site?

DBT DEVELOPMENT & DEPLOYMENT FLOW



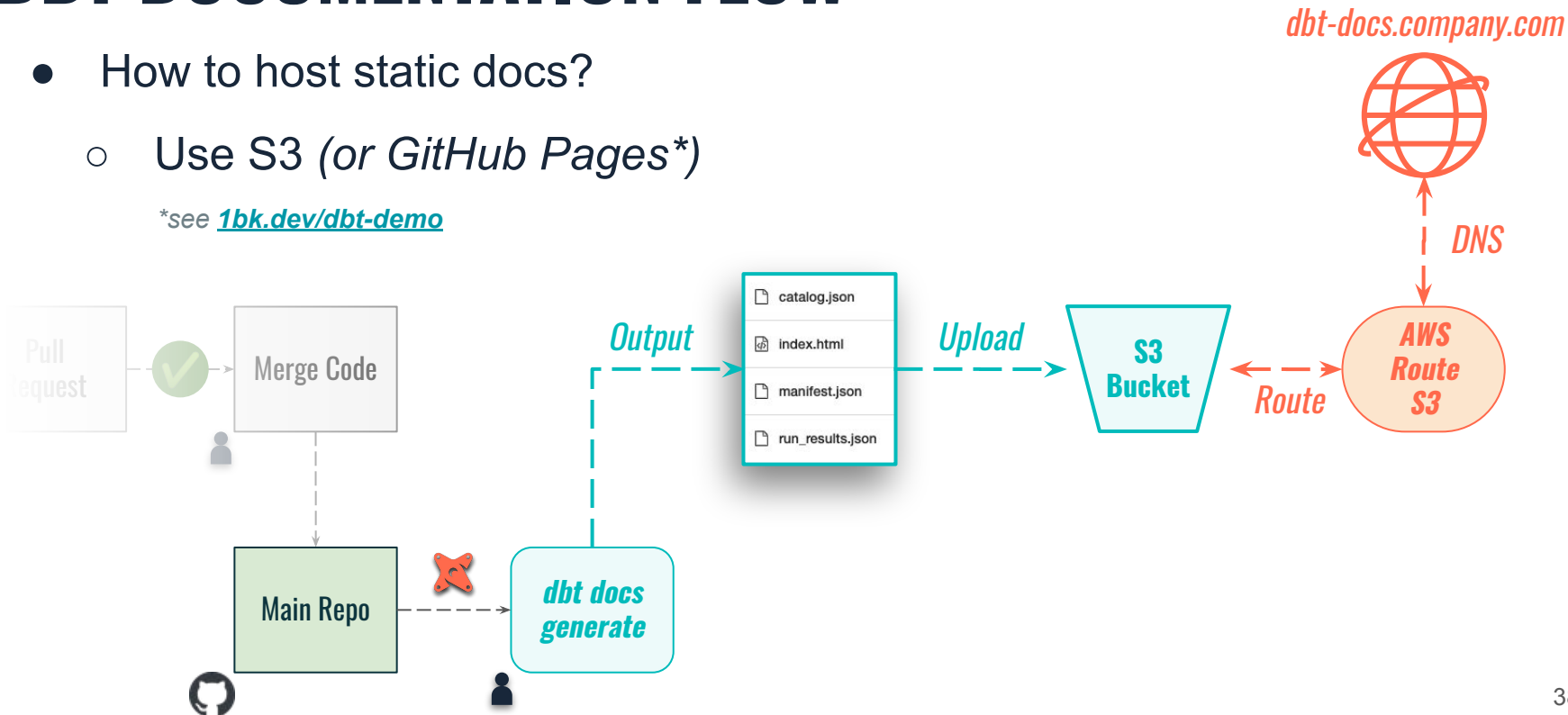
DBT EXECUTION FLOW



DBT DOCUMENTATION FLOW

- How to host static docs?
 - Use S3 (or GitHub Pages*)

*see 1bk.dev/dbt-demo



ATTEMPTING THE CHANGE

- It's time to get everyone else on-board!
- First, **presentation** to the company.
- Expect low initial buy-in, so invest more time...
- Host multiple hands-on **tutorial sessions**
- Set up Slack channels for support and sharing

MOTIVATION

WHAT IS DBT

PROBLEM-SOLUTION

PRODUCTIONISATION

ANALYTICS TRANSFORMED

ANALYTICS TRANSFORMED

BEFORE

- DS pass SQL to DE to run
- Disorganised code
- Messy and complex SQL
- Repetitive code
- Little or no documentation
- Lack proper framework

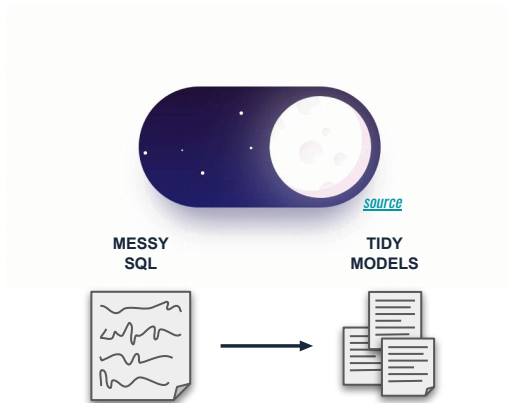
AFTER

- More shared ownership
- Git integration & version control
- Clear model structure & lineage
- DRY code with Models & Macros
- Sleek documentation site
- Clear analytic workflow

BUT But but...

BUT But but...

- Busy with BAU
- Time consuming to migrate
- Steep learning curve



CLI?

Git?

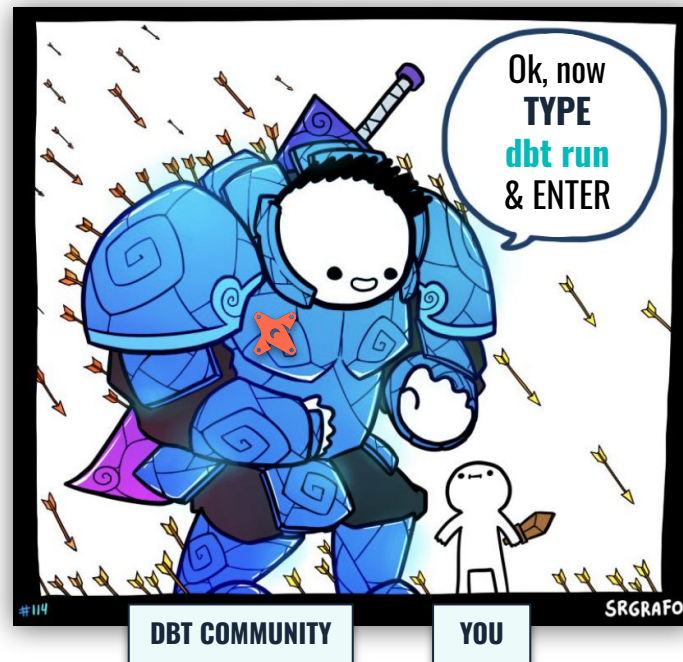


Version
Control?

YAML?

BUT But but...

- Busy with BAU
- Time consuming to migrate
- Steep learning curve
- Yes, the journey can be tough
... but not as bad as you think!



MONEYLION'S FIRST STEPS

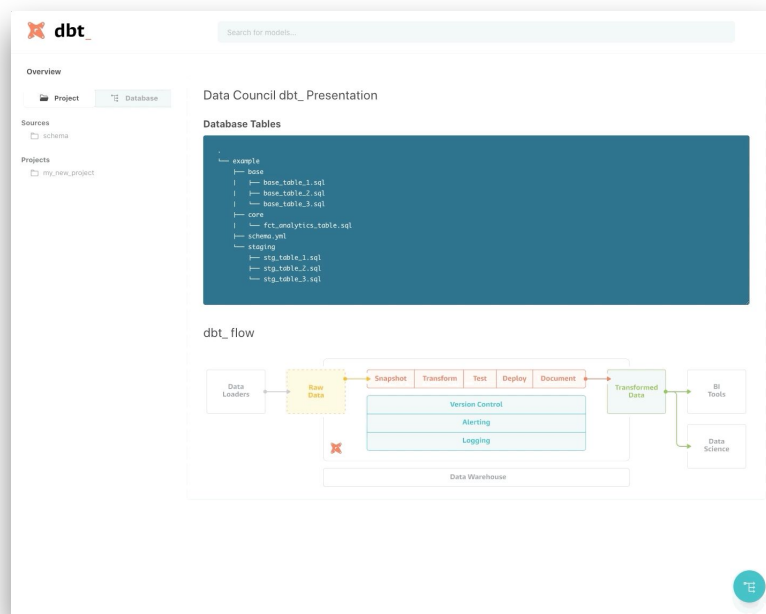
BEFORE

```

1 WITH employee AS
2 (
3   --SELECT DISTINCT id
4   --FROM (SELECT id
5   --FROM schema.subs
6   --WHERE LOWER(subtype) LIKE ('%employee%')
7   --OR id = '123123123123'
8   --UNION
9   --SELECT id
10  --FROM schema.users
11  --WHERE email LIKE '%@company.com')
12 )
13
14 req_final_cte AS
15 (
16   --SELECT id ..... AS req_id,
17   --user_id ..... AS user_id,
18   --created_at ..... AS laon_reqlied_at,
19   --effective_date ..... AS laon_effective_at,
20   --disbursed_at ..... AS laon_disbursed_at,
21   --"type" ..... AS release_type,
22   --status ..... AS high_level_status,
23   --failure_reason ..... AS req_failure_reason,
24   --receival_at ..... AS laon_receival_at,
25   --txsE
26   --WHEN laon_receival_at < CURRENT_DATE THEN TRUE
27   --ELSE FALSE
28   --END AS laon_due,
29   --original_maximum_eligible AS original_maximum_eligible,
30   --max_amount ..... AS laon_max_amount,
31   --min_amount ..... AS laon_min_amount,
32   --available_amount ..... AS laon_available_amount,
33   --txsE
34   --WHEN receival_at IS NULL AND laon_amount = 2 THEN 'A_TIER'
35   --ELSE tier_name = '' AND laon_amount > 2 THEN 'B_TIER'
36   --END AS laon_tier_name,
37   --laon_amount ..... AS laon_amount_on_req
38   --FROM schema.tx_req
39 )
40 tx_cdd_cte AS
41 (
42   --SELECT tx.id ..... AS laon_id,

```

AFTER



MIGRATED IN < 1 DAY

YOUR FIRST STEPS

- Git clone the Official Tutorial:

https://github.com/fishtown-analytics/jaffle_shop

- What you need:

- **Python 3.6 +**

- A database:

- **Postgres** (docker: https://hub.docker.com/_/postgres)

- Or others...

See <https://docs.getdbt.com/docs/supported-databases/>

SETUP

Install dbt

```
$ pip install dbt
Successfully installed dbt-0.xx.x
```

Setup and Test DB Connection

```
$ vim ../dbt/profiles.yml
$ dbt debug
...
Connection test: OK connection ok
```

Load CSV files into Database

```
$ dbt seed|
1 of 3 OK loaded seed file public.raw_customers..... [INSERT 100 in 0.48s]
2 of 3 OK loaded seed file public.raw_orders..... [INSERT 99 in 0.48s]
3 of 3 OK loaded seed file public.raw_payments..... [INSERT 113 in 0.46s]

Finished running 3 seeds in 1.36s.

Done. PASS=3 WARN=0 ERROR=0 SKIP=0 TOTAL=3
```

SETUP

Setup and Test DB Connection

```
profiles.yml
1  jaffle_shop:
2    outputs:
3      dev:
4        type: postgres
5        threads: 8
6        host: localhost
7        port: 5432
8        user: postgres
9        pass: docker
10       dbname: postgres
11       schema: public
12     target: dev
13
```


EXECUTE

Materialise Tables on Database

```
bash
$ dbt run
Concurrency: 8 threads (target='dev')
...
8 of 8 START table model public.dim_customers..... [RUN]
8 of 8 OK created table model public.dim_customers..... [SELECT 100 in 0.11s]

Finished running 3 view models, 5 table models in 1.29s.

Done. PASS=8 WARN=0 ERROR=0 SKIP=0 TOTAL=8
```

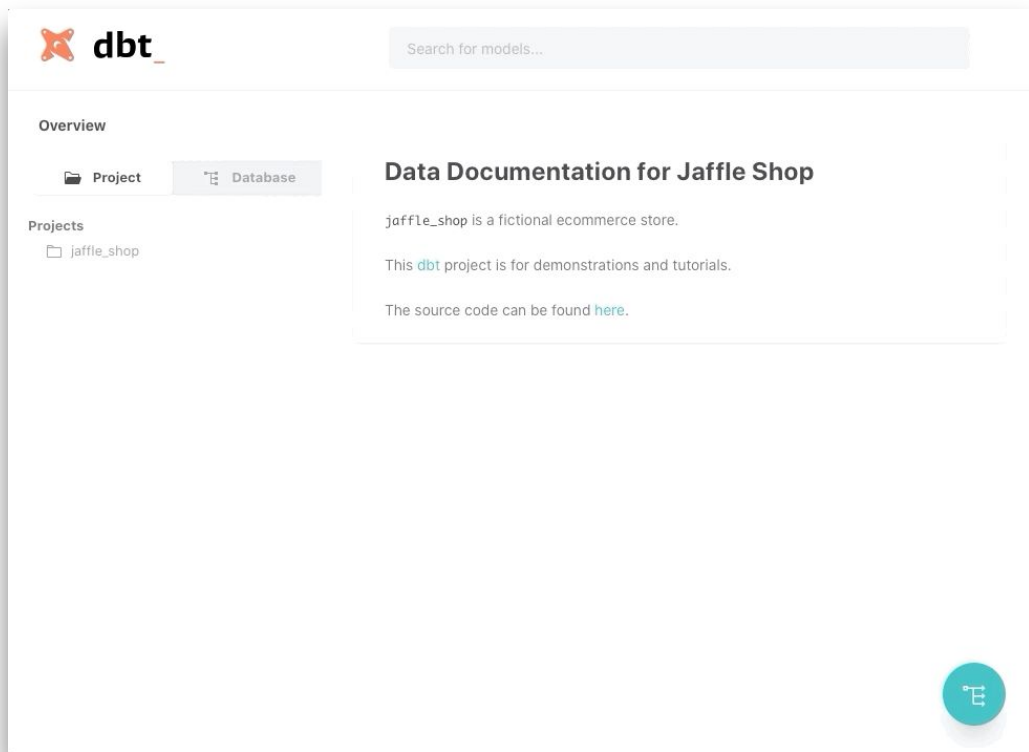
Generate Docs (HTML)

```
$ dbt docs generate
Building catalog
Catalog written to /Users/dbt/jaffle_shop/target/catalog.json
```

Host Docs Website (Locally)

```
$ dbt docs serve
Serving docs at 0.0.0.0:8888
To access from your browser, navigate to: http://localhost:8888
```

USE (DOCUMENTATION)



ONE MORE THING

OTHER AWESOME FEATURES

- Hooks ***
- Snapshots
- Testing *****
- Custom schema tests ***
- Source freshness checks ***
- Configuring models ***
- Model selection syntax *
- Custom Profiles Directory *****
- Seeds **
- Macros *****
- Environmental Variables ***
- Docs Blocks **
- Using tags *
- Packages

* = level of awesomeness (i.e. we use a lot in MoneyLion)

GREAT DBT RESOURCES

OFFICIAL RESOURCES

- [Tutorial](#) (fishtown-analytics/jaffle_shop)
- [Best Practices](#) ***
- [How we structure our dbt projects](#) ***
- [How we set up our computers for working on dbt projects](#)
- [How Monzo, the UK's favorite mobile-only bank, is rolling out dbt](#)

COMMUNITY

- [Join the dbt community](#) (Slack channel)! ***

PERSONAL

- Example dbt docs: [1bk.dev/dbt-demo](#)

END