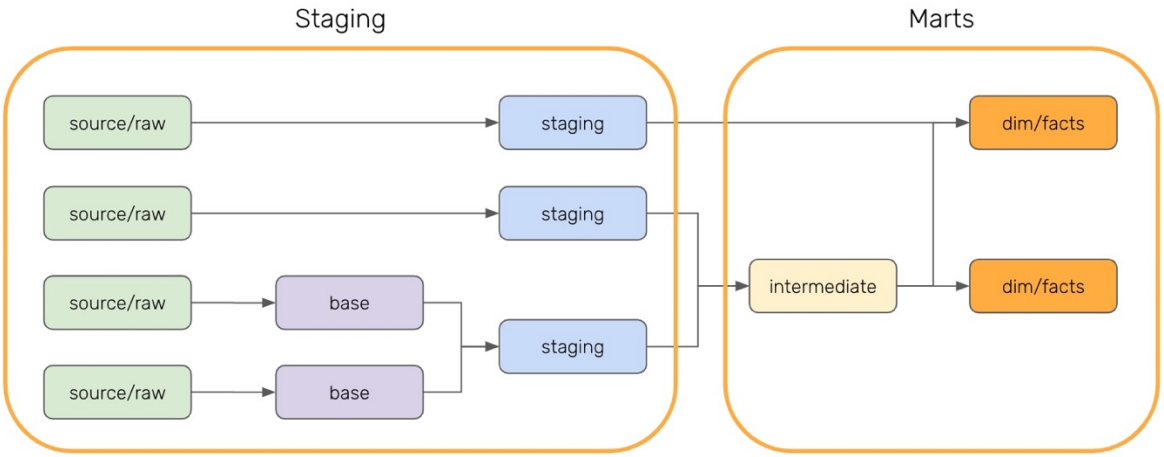
Week 6 : 06-analytics-engineering

Model Layers



Model layers are composed of 2 parts;

Staging part is the area of transforming data or source

Marts part is the area of joining data models preparing for dashboard

Project Processing

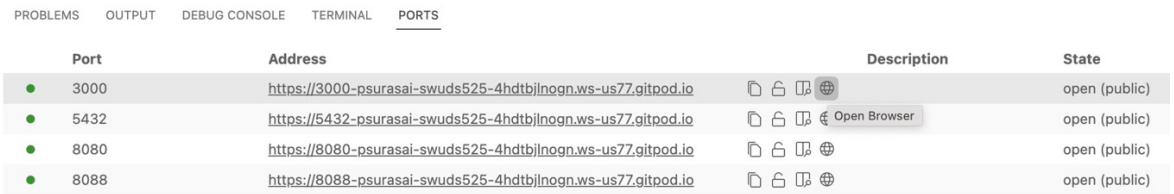
1. change directory to project 06-analystics-engineering
2. $ docker-compose up

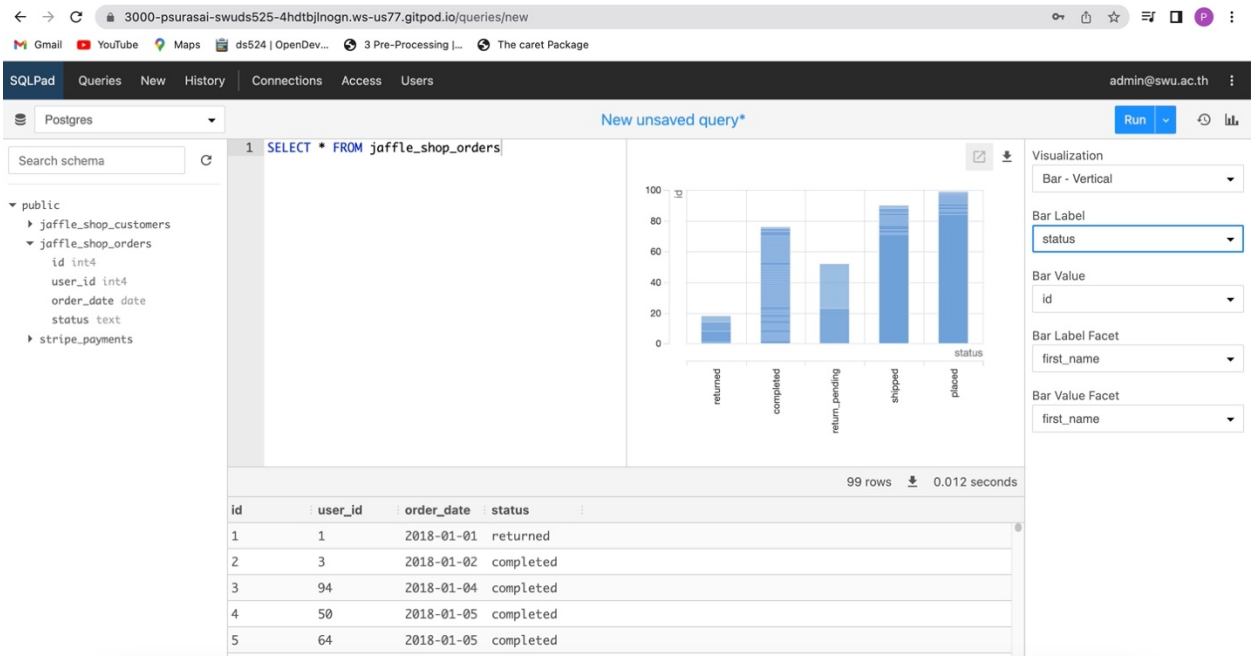
```
gitpod /workspace/SWU-DS525 (main) $ cd 06-analystics-engineering/
gitpod /workspace/SWU-DS525/06-analystics-engineering (main) $ docker-compose up
```

3. Open SQLPad browser localhost:3000

   SQLPAD_ADMIN: admin@swu.ac.th

   SQLPAD_ADMIN_PASSWORD: admin





4. Create python virtual environment and install required applications

   $ cd 06-analytics-engineering/

   $ python -m venv ENV

   $ source ENV/bin/activate

   $ pip install -r requirements.txt

```
● gitpod /workspace/SWU-DS525 (main) $ cd 06-analytics-engineering/
● gitpod /workspace/SWU-DS525/06-analytics-engineering (main) $ python -m venv ENV
● gitpod /workspace/SWU-DS525/06-analytics-engineering (main) $ source ENV/bin/activate
● (ENV) gitpod /workspace/SWU-DS525/06-analytics-engineering (main) $ pip install -r requirements.txt
Collecting agate==1.6.3
  Downloading agate-1.6.3-py2.py3-none-any.whl (100 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100.5/100.5 KB 5.5 MB/s eta 0:00:00
Collecting attrs==22.1.0
  Downloading attrs-22.1.0-py2.py3-none-any.whl (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.8/58.8 KB 12.3 MB/s eta 0:00:00
Collecting Babel==2.10.3
```

5. Create dbt project

   $ dbt (to check if it is working)

```
(ENV) gitpod /workspace/SWU-DS525/06-analystics-engineering/jaffle (main) $ dbt
usage: dbt [-h] [--version] [-r RECORD_TIMING_INFO] [-d] [--log-format {text,json,default}] [--no-write-json]
           [--use-colors | --no-use-colors] [--printer-width PRINTER_WIDTH] [--warn-error] [--no-version-check]
           [--partial-parse | --no-partial-parse] [--use-experimental-parser] [--no-static-parser] [--profiles-dir PROFILES_DIR]
           [--no-anonymous-usage-stats] [-x] [--event-buffer-size EVENT_BUFFER_SIZE] [-q] [--no-print]
           [--cache-selected-only | --no-cache-selected-only]
           {docs,source,init,clean,debug,deps,list,ls,build,snapshot,run,compile,parse,test,seed,run-operation} ...

An ELT tool for managing your SQL transformations and data models. For more documentation on these commands, visit: docs.getdbt.com

optional arguments:
  -h, --help            show this help message and exit
  --version             Show version information
  -r RECORD_TIMING_INFO, --record-timing-info RECORD_TIMING_INFO
                        When this option is passed, dbt will output low-level timing stats to the specified file. Example: `--record-
                        timing-info output.profile`
  -d, --debug           Display debug logging during dbt execution. Useful for debugging and making bug reports.
  --log-format {text,json,default}
                        Specify the log format, overriding the command's default.
  --no-write-json       If set, skip writing the manifest and run_results.json files to disk
  --use-colors          Colorize the output DBT prints to the terminal. Output is colorized by default and may also be set in a
                        profile or at the command line. Mutually exclusive with --no-use-colors
  --no-use-colors       Do not colorize the output DBT prints to the terminal. Output is colorized by default and may also be set in a
                        profile or at the command line. Mutually exclusive with --use-colors
  --printer-width PRINTER_WIDTH
                        Sets the width of terminal output
  --warn-error          If dbt would normally warn, instead raise an exception. Examples include --models that selects nothing,
                        deprecations, configurations with no associated models, invalid test configurations, and missing sources/refs
                        in tests.
  --no-version-check    If set, skip ensuring dbt's version matches the one specified in the dbt_project.yml file ('require-dbt-
                        version')
  --partial-parse       Allow for partial parsing by looking for and writing to a pickle file in the target directory. This overrides
                        the user configuration file.
  --no-partial-parse    Disallow partial parsing. This overrides the user configuration file.
  --use-experimental-parser
                        Enables experimental parsing features.
  --no-static-parser    Disables the static parser.
  --profiles-dir PROFILES_DIR
```

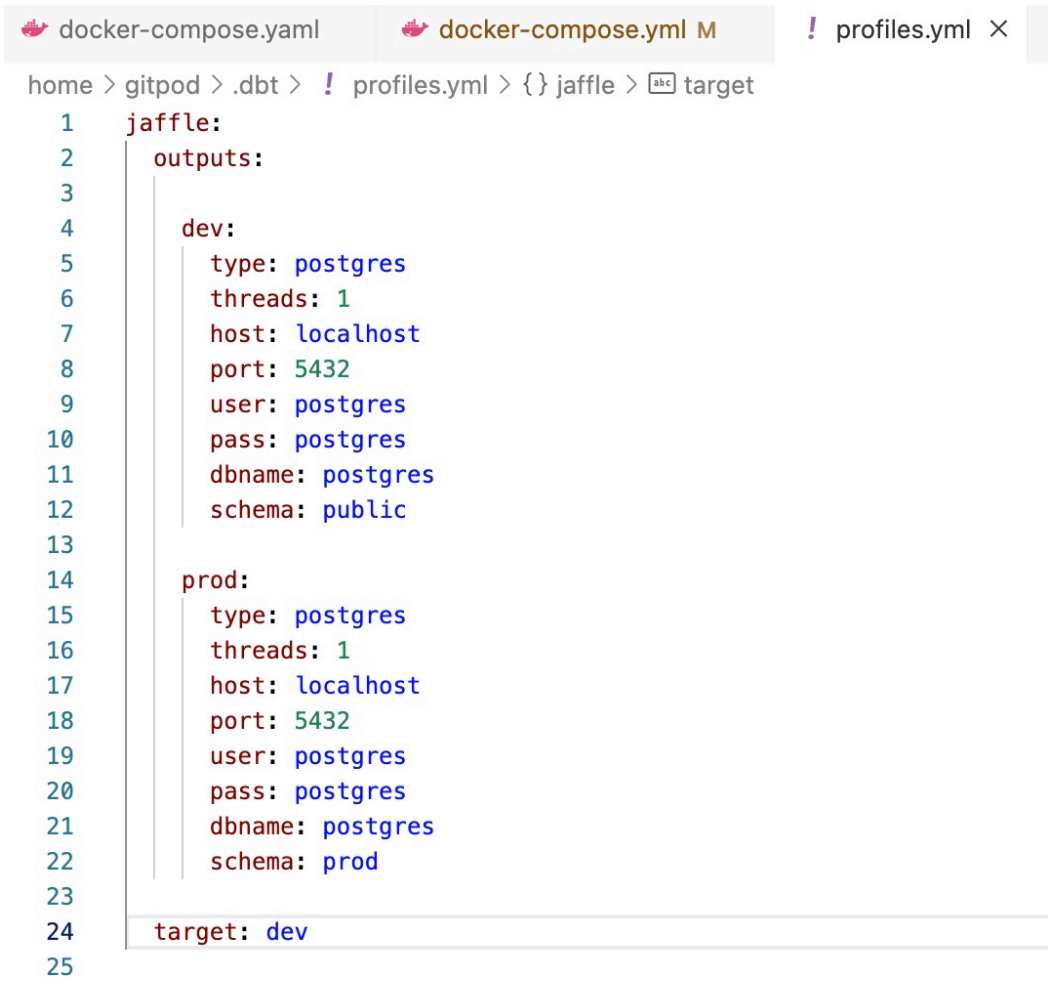
$ dbt init -> Name the file and select database (type “1” for postgres in this project)

```
(ENV) gitpod /workspace/swu-ds525/06-analytics-engineering (main) $ dbt init
10:08:03  Running with dbt=1.2.0
10:08:03  Creating dbt configuration folder at /home/gitpod/.dbt
Enter a name for your project (letters, digits, underscore): jaffle
Which database would you like to use?
[1] postgres

(Don't see the one you want? https://docs.getdbt.com/docs/available-adapters)

Enter a number: 1
```

6. Set profile

   $ code ~/.dbt/profiles.yml

```
(ENV) gitpod /workspace/SWU-DS525/06-analystics-engineering/jaffle (main) $ code ~/.dbt/profiles.yml
```



7. Create folders marts/sales and staging/jaffle for containing following files

7.1 Staging

Setup source by

7.1.1 Create stg_jaffle__customers.sql to extract data from customers table in staging/jaffle

```
with
source as (
 select * from {{ source('jaffle', 'jaffle_shop_customers') }}
)
, final as (
 select
 id
```

```
 , first_name || ' ' || last_name as name
 from source
)
select * from final
```

7.1.2 Create stg_jaffle__orders.sql to extract data from orders table in staging/jaffle

```
with
source as (
 select * from {{ source('jaffle', 'jaffle_shop_orders') }}
)
, final as (
 select
 id
 , user_id
 , order_date
 , status
 from source
)
select * from final
```

7.1.3 Create stg_jaffle__stripe_payments.sql to extract data from stripe_payments in staging/jaffle

```
with
source as (
 select * from {{ source('jaffle', 'stripe_payments') }}
)
, final as (
 select
 id
```

```
 , order_id
 , payment_method
 , amount
 , status
 , created
 from source
)
select * from final
```

7.1.4 Create stg_models.yml to write Doc from staging ***** in staging/jaffle

```
version: 2
models:
 - name: stg_jaffle__customers
 description: Staging model for customers
 columns:
 - name: id
 tests:
 - unique
 - not_null
 - name: name
 - name: stg_jaffle__orders
 description: Staging model for orders
 columns:
 - name: id
 - name: user_id
 - name: order_date
 - name: status
 - name: stg_jaffle__stripe_payments
 description: Staging model for Stripe payments
 columns:
```

```
 - name: id
 - name: order_id
 - name: payment_method
 - name: amount
 - name: status
 - name: created
```

7.2 Marts

7.2.1 Inport complete_orders.sql to marts/sales by writing the following code

```
with
int_orders_customers_joined as (
 select * from {{ ref('int_orders_customers_joined') }}
)
, final as (
 select
 order_id
 , order_date
 , order_status
 , customer_name
 from int_orders_customers_joined
 where order_status = 'completed'
)
select * from final
```

7.2.2 Create pending_orders.sql ใน marts/sales

```
with
int_orders_customers_joined as (
 select * from {{ ref('int_orders_customers_joined') }}
)
, final as (
```

```
 Select

order_id
 , order_date
 , order_status
 , customer_name
 from int_orders_customers_joined
 where order_status = 'pending'
)
select * from final
```

7.2.3 Create Folder intermediate in marts/sales and file int_orders_customers_joined.sql

```
with
orders as (
 select * from {{ ref('stg_jaffle__orders') }}
)
, customers as (
 select * from {{ ref('stg_jaffle__customers') }}
)
, final as (
 select
 o.id as order_id
 , o.order_date
 , o.status as order_status
 , c.name as customer_name
 from orders as o
 join customers as c
 on
```

o.user_id = c.id

)

select * from final

7.2.4 Create exposures.yml in mart/sales to prepare a dashboard

version: 2

exposures:

Check assert_completed_orders_should_have_only_completed_status.sql in

test

select

status

from "postgres"."public"."completed_orders"

where status != 'completed'

7.3 Run DBT

To create models

$ dbt run

```
(ENV) gitpod /workspace/SWU-DS525/06-analystics-engineering/jaffle (main) $ dbt run
15:14:27  Running with dbt=1.2.0
15:14:27  Found 8 models, 9 tests, 0 snapshots, 0 analyses, 256 macros, 0 operations, 0 seed files, 3 sources, 1 exposu
re, 0 metrics
15:14:27
15:14:27  Concurrency: 1 threads (target='dev')
15:14:27
15:14:27  1 of 8 START table model public.my_first_dbt_model ............................. [RUN]
15:14:27  1 of 8 OK created table model public.my_first_dbt_model ........................ [SELECT 1 in 0.12s]
15:14:27  2 of 8 START view model public.stg_jaffle__customers ........................... [RUN]
15:14:27  2 of 8 OK created view model public.stg_jaffle__customers ...................... [CREATE VIEW in 0.05s]
15:14:27  3 of 8 START view model public.stg_jaffle__orders .............................. [RUN]
15:14:27  3 of 8 OK created view model public.stg_jaffle__orders ......................... [CREATE VIEW in 0.04s]
15:14:27  4 of 8 START view model public.stg_jaffle__stripe_payments ..................... [RUN]
15:14:27  4 of 8 OK created view model public.stg_jaffle__stripe_payments ................ [CREATE VIEW in 0.03s]
15:14:27  5 of 8 START view model public.my_second_dbt_model ............................. [RUN]
15:14:27  5 of 8 OK created view model public.my_second_dbt_model ........................ [CREATE VIEW in 0.03s]
15:14:27  6 of 8 START view model public.int_orders_customers_joined ..................... [RUN]
15:14:27  6 of 8 OK created view model public.int_orders_customers_joined ................ [CREATE VIEW in 0.03s]
15:14:27  7 of 8 START view model public.completed_orders ................................ [RUN]
15:14:27  7 of 8 OK created view model public.completed_orders ........................... [CREATE VIEW in 0.03s]
15:14:27  8 of 8 START view model public.pending_orders .................................. [RUN]
15:14:27  8 of 8 OK created view model public.pending_orders ............................. [CREATE VIEW in 0.03s]
15:14:27
15:14:27  Finished running 1 table model, 7 view models in 0 hours 0 minutes and 0.50 seconds (0.50s).
15:14:27
15:14:27  Completed successfully
15:14:27
15:14:27  Done. PASS=8 WARN=0 ERROR=0 SKIP=0 TOTAL=8
(ENV) gitpod /workspace/SWU-DS525/06-analystics-engineering/jaffle (main) $
```

To test models

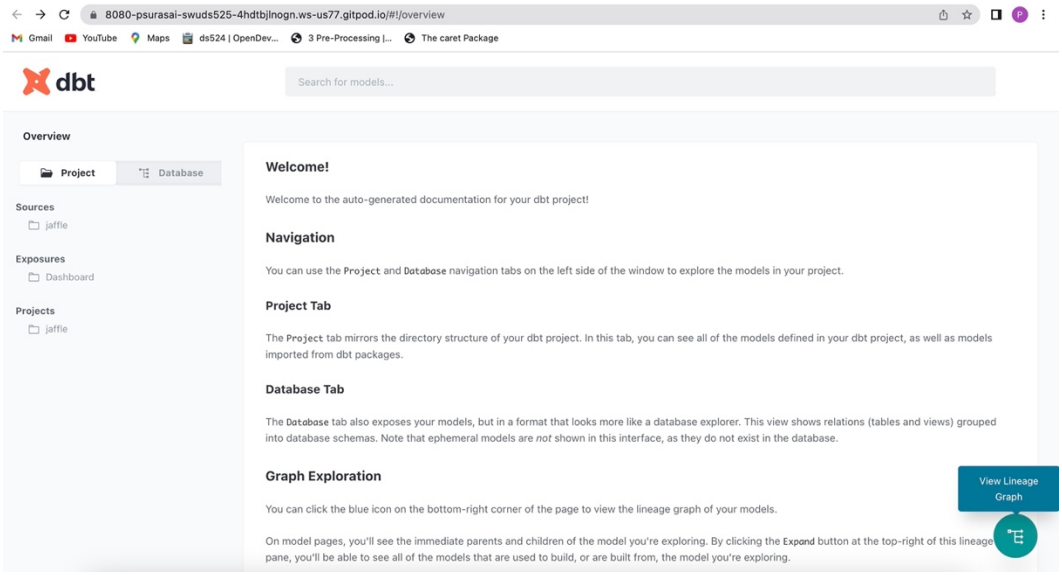
$ dbt test

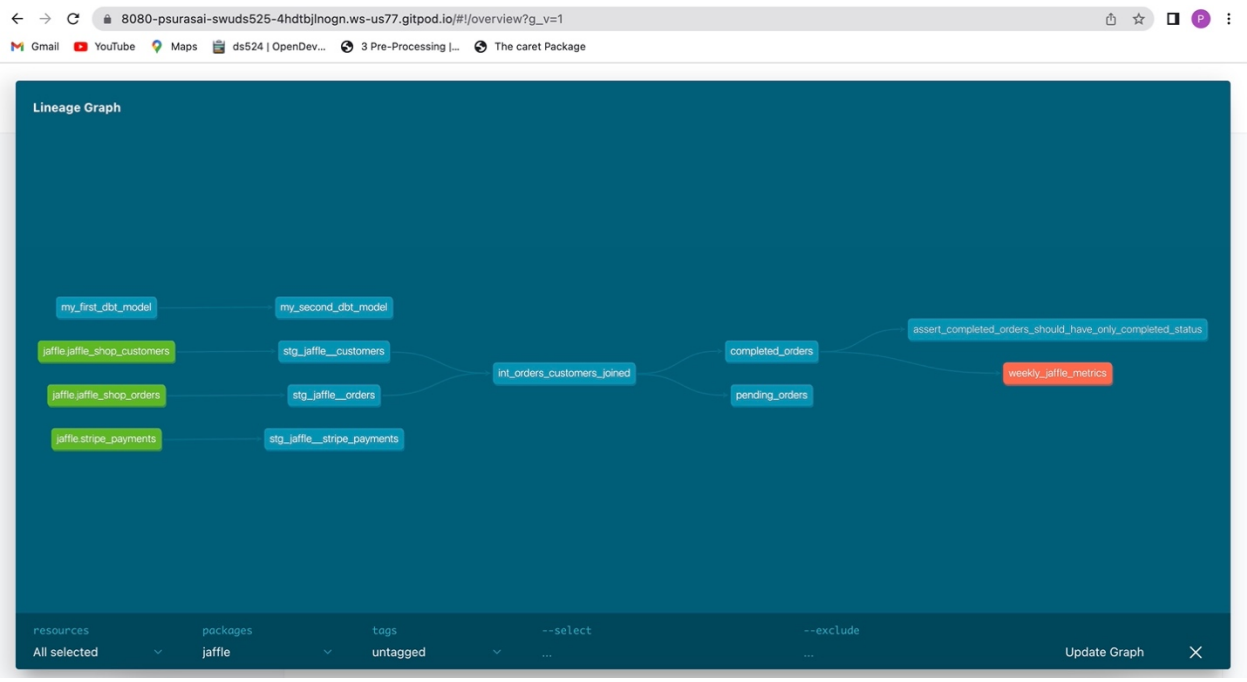
```
15:14:27  Done. PASS=8 WARN=0 ERROR=0 SKIP=0 TOTAL=8
(ENV) gitpod /workspace/SWU-DS525/06-analytics-engineering/jaffle (main) $ dbt test
15:15:12  Running with dbt=1.2.0
15:15:12  Found 8 models, 9 tests, 0 snapshots, 0 analyses, 256 macros, 0 operations, 0 seed files, 3 sources, 1 exposu
re, 0 metrics
15:15:12
15:15:12  Concurrency: 1 threads (target='dev')
15:15:12
15:15:12  1 of 9 START test accepted_values_stg_jaffle__orders_status__placed__shipped__return_pending__returned__compl
eted  [RUN]
15:15:13  1 of 9 PASS accepted_values_stg_jaffle__orders_status__placed__shipped__return_pending__returned__completed
[PASS in 0.05s]
15:15:13  2 of 9 START test assert_completed_orders_should_have_only_completed_status .... [RUN]
15:15:13  2 of 9 PASS assert_completed_orders_should_have_only_completed_status ......... [PASS in 0.02s]
15:15:13  3 of 9 START test not_null_my_first_dbt_model_id ............................... [RUN]
15:15:13  3 of 9 PASS not_null_my_first_dbt_model_id ..................................... [PASS in 0.03s]
15:15:13  4 of 9 START test not_null_my_second_dbt_model_id .............................. [RUN]
15:15:13  4 of 9 PASS not_null_my_second_dbt_model_id .................................... [PASS in 0.02s]
15:15:13  5 of 9 START test not_null_stg_jaffle__customers_id ............................ [RUN]
15:15:13  5 of 9 PASS not_null_stg_jaffle__customers_id .................................. [PASS in 0.02s]
15:15:13  6 of 9 START test relationships_stg_jaffle__orders_user_id__id__ref_stg_jaffle__customers_  [RUN]
15:15:13  6 of 9 PASS relationships_stg_jaffle__orders_user_id__id__ref_stg_jaffle__customers_  [PASS in 0.03s]
15:15:13  7 of 9 START test unique_my_first_dbt_model_id .................................. [RUN]
15:15:13  7 of 9 PASS unique_my_first_dbt_model_id ....................................... [PASS in 0.02s]
15:15:13  8 of 9 START test unique_my_second_dbt_model_id ................................. [RUN]
15:15:13  8 of 9 PASS unique_my_second_dbt_model_id ...................................... [PASS in 0.02s]
15:15:13  9 of 9 START test unique_stg_jaffle__customers_id .............................. [RUN]
15:15:13  9 of 9 PASS unique_stg_jaffle__customers_id .................................... [PASS in 0.02s]
15:15:13
15:15:13  Finished running 9 tests in 0 hours 0 minutes and 0.35 seconds (0.35s).
15:15:13
15:15:13  Completed successfully
15:15:13
15:15:13  Done. PASS=9 WARN=0 ERROR=0 SKIP=0 TOTAL=9
(ENV) gitpod /workspace/SWU-DS525/06-analytics-engineering/jaffle (main) $
```

To view docs (on Gitpod)

$ dbt docs generate

$ dbt docs serve




8080-psurasai-swuds525-4hdtbjlnogn.ws-us77.gitpod.io/#!/overview?g_v=1
Gmail
YouTube
Maps
ds524 | OpenDev...
3 Pre-Processing |...
The caret Package
Lineage Graph
my_first_dbt_model
my_second_dbt_model
jaffle.jaffle_shop_customers
stg_jaffle__customers
jaffle.jaffle_shop_orders
stg_jaffle__orders
jaffle.stripe_payments
stg_jaffle__stripe_payments
int_orders_customers_joined
completed_orders
pending_orders
assert_completed_orders_should_have_only_completed_status
weekly_jaffle_metrics
resources
All selected
packages
jaffle
tags
untagged
--select
...
--exclude
...
Update Graph