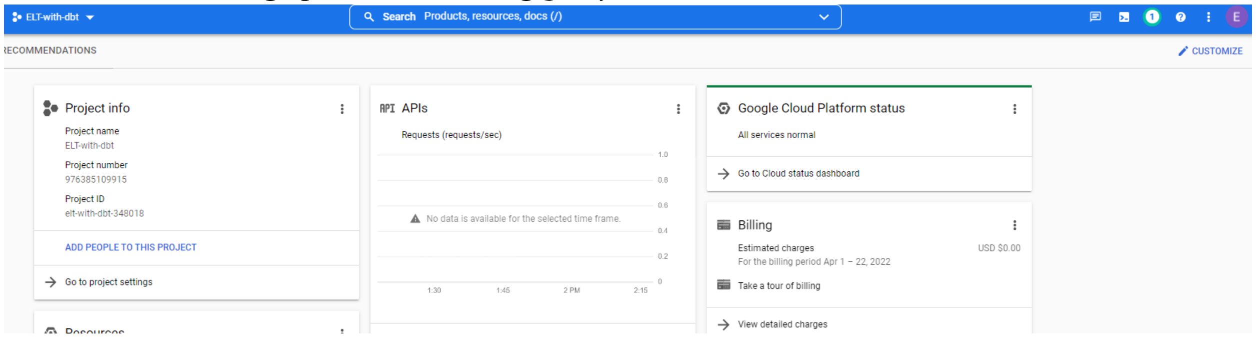
Esme Gonzalez
CIS 4400 Homework #3
ELT with dbt (Data Build Tool)

1. Getting started tutorials

a. *Setting up and connect BigQuery*

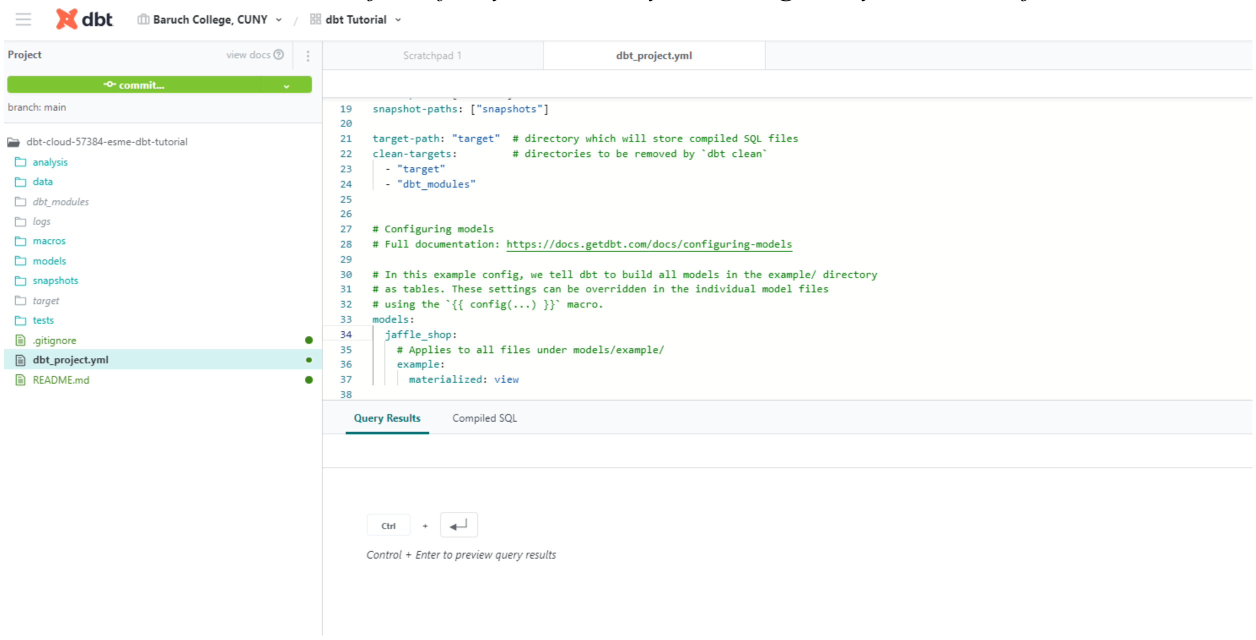


Project name
ELT-with-dbt
Project number
976385109915
Project ID
elt-with-dbt-348018

Estimated charges USD $0.00
For the billing period Apr 1 – 22, 2022

Creating a BigQuery Project using the instructions.

b. *Create a Project after you commit your changes in your dbt Project*



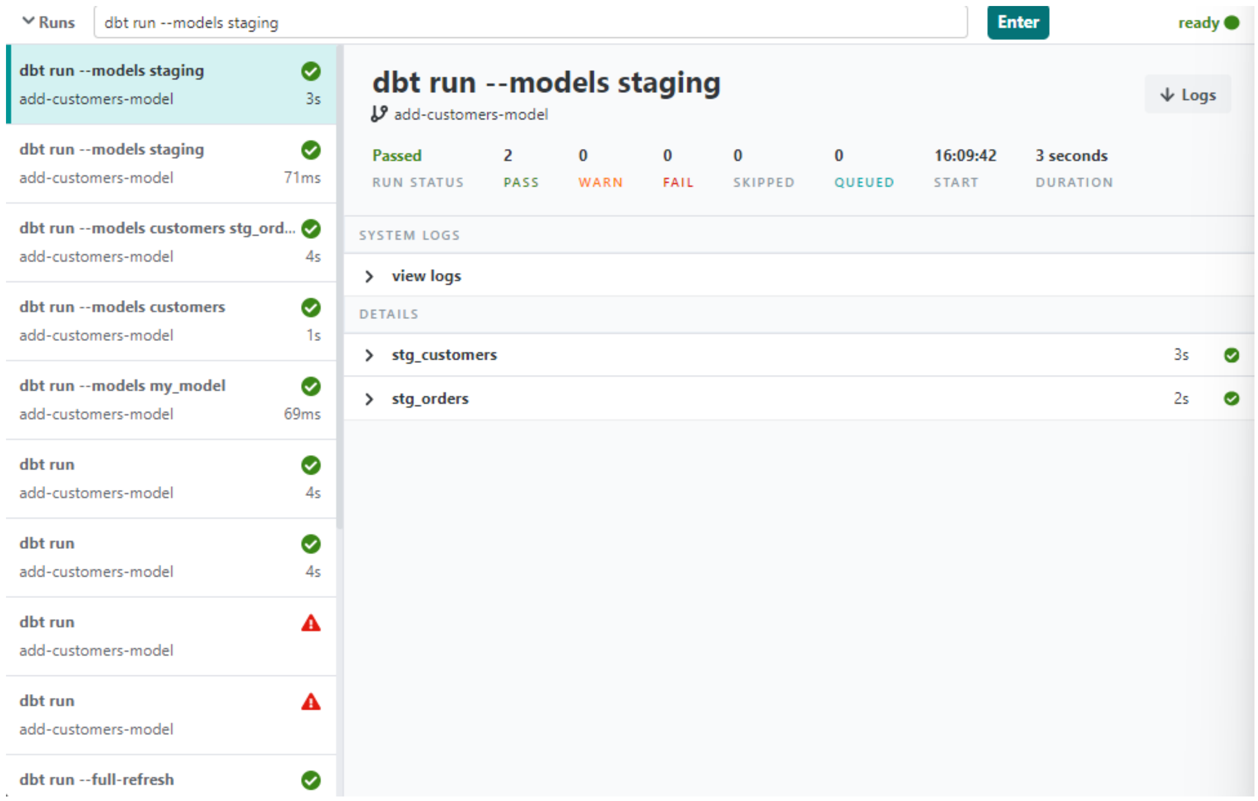
```
snapshot-paths: ["snapshots"]

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


# Configuring models
# Full documentation: https://docs.getdbt.com/docs/configuring-models

# In this example config, we tell dbt to build all models in the example/ directory
# as tables. These settings can be overridden in the individual model files
# using the `{{ config(...) }}` macro.
models:
  jaffle_shop:
    # Applies to all files under models/example/
    example:
      materialized: view

```

Created a dbt cloud account and connected it to Bigquery by following the instructions.

c. *Build Your First Model after you run the staging models.*

Following the instructions I built my first model by inputting the sql code below. Then creating and inputting the staging models for the sql files. Then run the dbt.

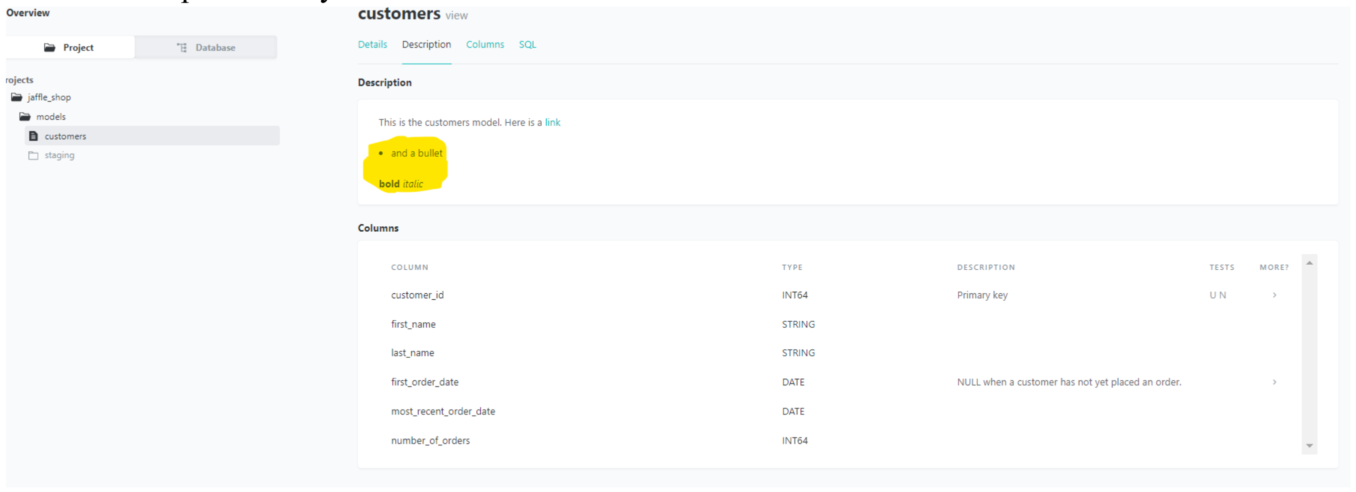
```
with customers as (select id as customer_id, first_name, last_name
  from `dbt-tutorial`.jaffle_shop.customers),
orders as (select id as order_id, user_id as customer_id, order_date, status
  from `dbt-tutorial`.jaffle_shop.orders),
customer_orders as (select customer_id, min(order_date) as first_order_date, max(order_date)
as most_recent_order_date, count(order_id) as number_of_orders
  from orders group by 1),
final as (select customers.customer_id, customers.first_name, customers.last_name,
customer_orders.first_order_date, customer_orders.most_recent_order_date,
coalesce(customer_orders.number_of_orders, 0) as number_of_orders
  from customers
  left join customer_orders using (customer_id))
select * from final
```



```
Runs   dbt run --models staging   Enter   ready

dbt run --models staging                    3s
add-customers-model
dbt run --models staging                    71ms
add-customers-model
dbt run --models customers stg_ord...       4s
add-customers-model
dbt run --models customers                  1s
add-customers-model
dbt run --models my_model                   69ms
add-customers-model
dbt run                                     4s
add-customers-model
dbt run                                     4s
add-customers-model
dbt run
add-customers-model
dbt run
add-customers-model
dbt run --full-refresh

dbt run --models staging
add-customers-model                                              Logs

Passed      2     0     0     0        0       16:09:42  3 seconds
RUN STATUS  PASS  WARN  FAIL  SKIPPED  QUEUED  START     DURATION

SYSTEM LOGS
> view logs
DETAILS
> stg_customers   3s
> stg_orders      2s
```

d. *Test and Document your project after you use the docs block to add a Markdown description to your model.*

Here, I will be inputting dbt tests and dbt docs generated for the customers table. Where I can edit the description on my tables.



Description

This is the customers model. Here is a link

- and a bullet

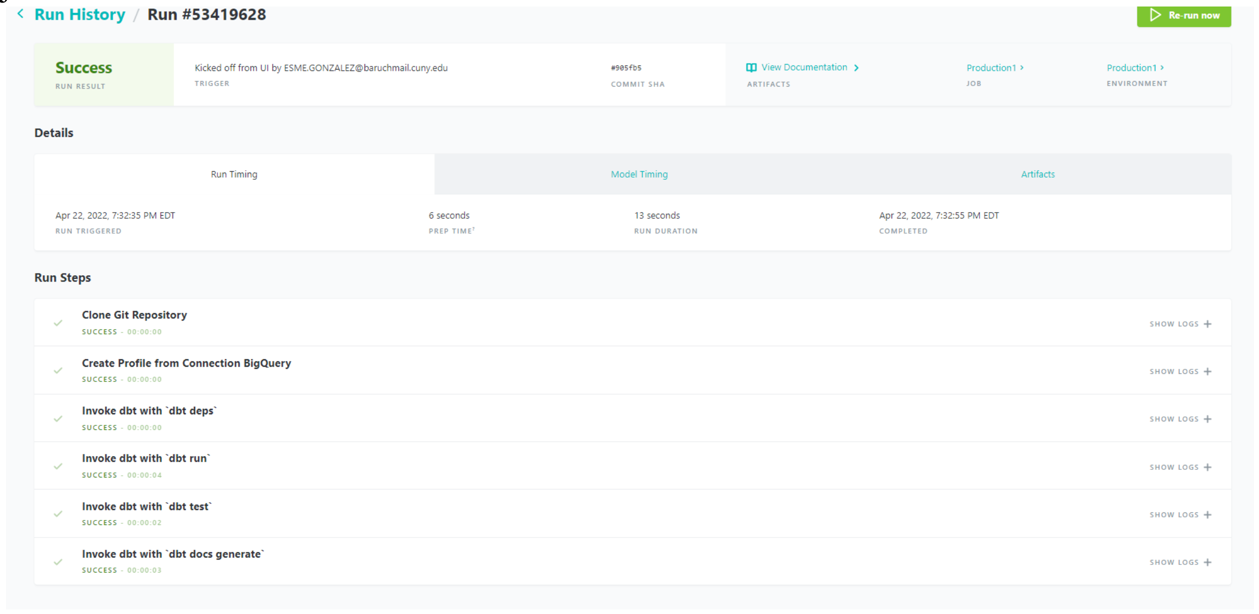
bold *italic*

Columns

COLUMN	TYPE	DESCRIPTION	TESTS	MORE?
customer_id	INT64	Primary key	U N	›
first_name	STRING			
last_name	STRING			
first_order_date	DATE	NULL when a customer has not yet placed an order.		›
most_recent_order_date	DATE			
number_of_orders	INT64			

e. *Deploy your project after you create and run a job.*

Here, I will create a deployment that is different from my original repository. Then run the dbt job.



Success	Kicked off from UI by ESME.GONZALEZ@baruchmail.cuny.edu	#905fb5	View Documentation ›	Production1 ›	Production1 ›
RUN RESULT	TRIGGER	COMMIT SHA	ARTIFACTS	JOB	ENVIRONMENT

Details

Run Timing | Model Timing | Artifacts

Apr 22, 2022, 7:32:35 PM EDT	6 seconds	13 seconds	Apr 22, 2022, 7:32:55 PM EDT
RUN TRIGGERED	PREP TIME	RUN DURATION	COMPLETED

Run Steps

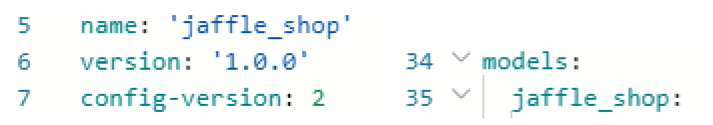
✓ **Clone Git Repository** SUCCESS - 00:00:00	SHOW LOGS +
✓ **Create Profile from Connection BigQuery** SUCCESS - 00:00:00	SHOW LOGS +
✓ **Invoke dbt with `dbt deps`** SUCCESS - 00:00:00	SHOW LOGS +
✓ **Invoke dbt with `dbt run`** SUCCESS - 00:00:04	SHOW LOGS +
✓ **Invoke dbt with `dbt test`** SUCCESS - 00:00:02	SHOW LOGS +
✓ **Invoke dbt with `dbt docs generate`** SUCCESS - 00:00:03	SHOW LOGS +

<u>The Four Courses</u>

Course one!!

I will be using bigquery and dbt for the four courses. I will also reference the code below
select * from `dbt-tutorial.jaffle_shop.customers`;
select * from `dbt-tutorial.jaffle_shop.orders`;
select * from `dbt-tutorial.stripe.payment`;

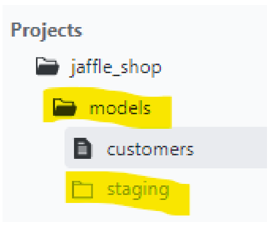
- ➔ **Models – Practice:**
  - ◆ <u>Quick Project Polishing</u>
    - ● In my project, I change the name of my project from my_new_project to jaffle_shop (line 5 AND 35)



```
5   name: 'jaffle_shop'
6   version: '1.0.0'
7   config-version: 2
```

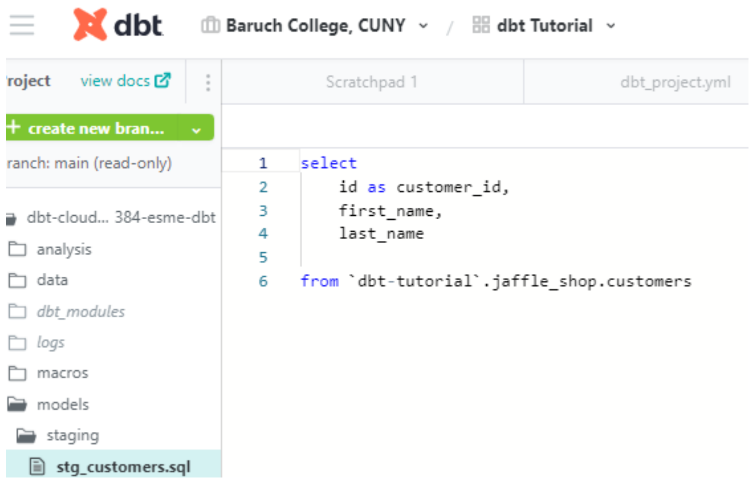
```
34  models:
35    jaffle_shop:
```

  - ◆ <u>Staging Models</u>
    - ● Shown below, I created a new folder called staging/jaffle_shop directory



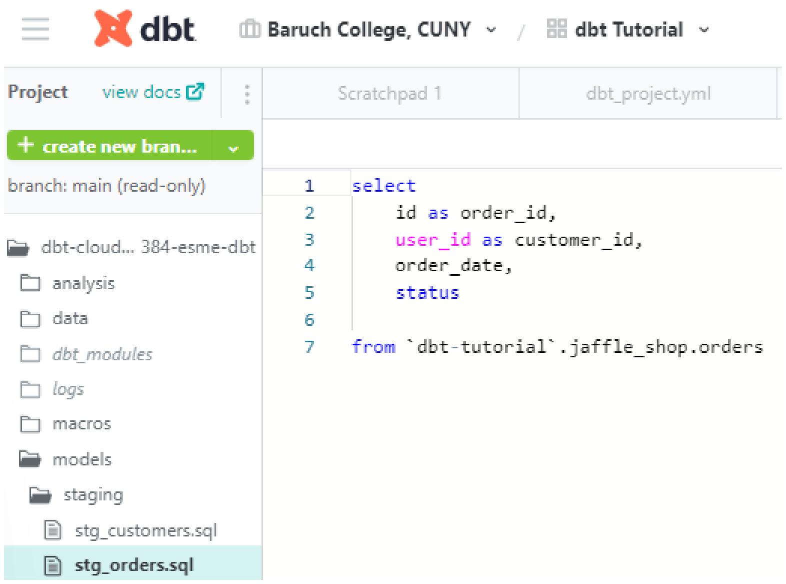
in my models folder for my project.

    - ● In the screenshot below, I create a stg_customers.sql model for the .jaffle_shop.customers.



```
select
    id as customer_id,
    first_name,
    last_name

from `dbt-tutorial`.jaffle_shop.customers
```

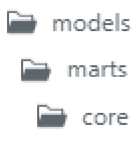
- After, I create a stg_orders.sql model for .jaffle_shop.orders



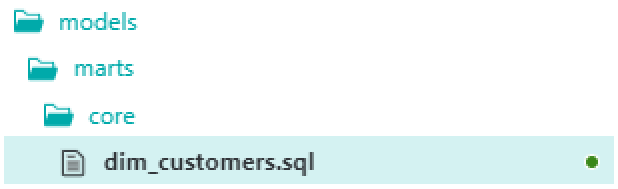
```
select
    id as order_id,
    user_id as customer_id,
    order_date,
    status

from `dbt-tutorial`.jaffle_shop.orders
```

◆ Mart Models

- For the mart models I will create a marts/core directory in my models folder. By clicking the three dots of the models folder and adding a new folder name marts/core.

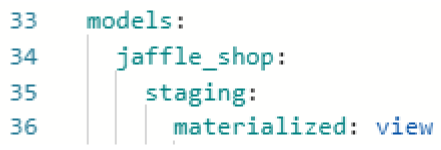


- To create a dim_customers.sql model. I will click the three dots from the core folder to create a new file called dim_customers.sql.



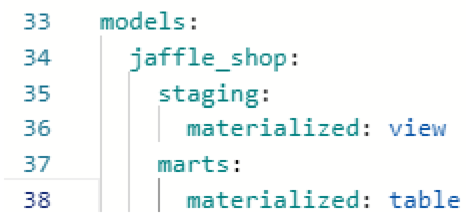
◆ Configure your materializations

- In my dbt_project.yml file, I will configure the staging directory to be materialized as views by typing it.



```
models:
  jaffle_shop:
    staging:
      materialized: view
```

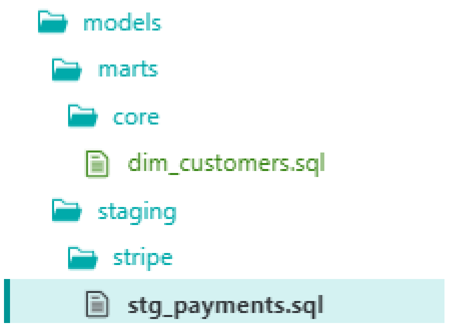
- In my dbt_project.yml file, I will configure the marts directory to be materialized as tables by typing and indenting correctly. The reason why I did not indent marts more is because marts is not in the staging folder, that is why I would get an error if I indent it more. But the screenshot works correctly by viewing the staging folder and doing a table in the marts folder.



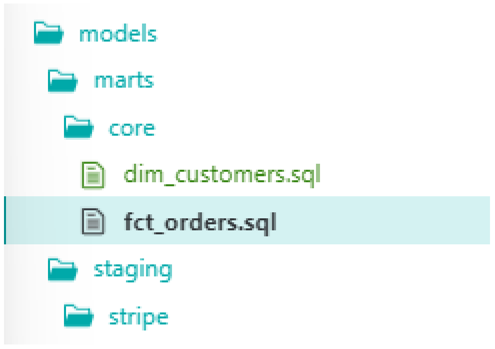
```
models:
  jaffle_shop:
    staging:
      materialized: view
    marts:
      materialized: table
```

◆ <u>Building a fct_orders Model</u>

- I then create a stg_payments.sql model in models/staging/stripe.



- Then I create a fct_orders.sql model with the following fields and put it in the marts/core directory.



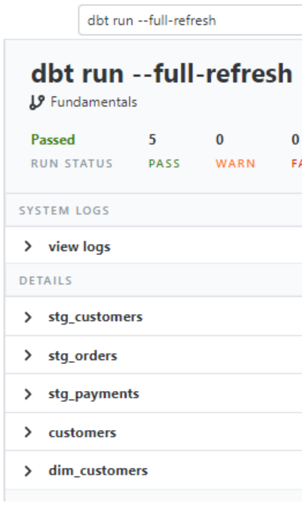
➔ **Models – Exemplar**

◆ <u>Self-check stg_payments, orders, customers</u>

- I will input my sql in the stripes folder with the one called the stg_payments. I would change my from raw.stripe.payment to 'dbt-tutorial'.stripe.payment instead. I would have to do this to the orders sql and customers sql too. When running my dbt I received an error but once I put the following:

```
select * from `dbt-tutorial.jaffle_shop.customers`;
select * from `dbt-tutorial.jaffle_shop.orders`;
select * from `dbt-tutorial.stripe.payment`;
```

In my bigquery and run a dbt full refresh. Where the dbt runs correctly.


dbt run --full-refresh

dbt run --full-refresh

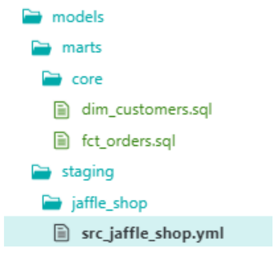
Fundamentals

Passed 5 0 0
RUN STATUS PASS WARN F

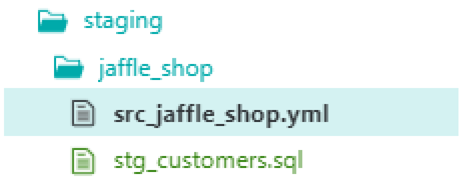
SYSTEM LOGS
view logs
DETAILS
stg_customers
stg_orders
stg_payments
customers
dim_customers

➔ Sources – Practice

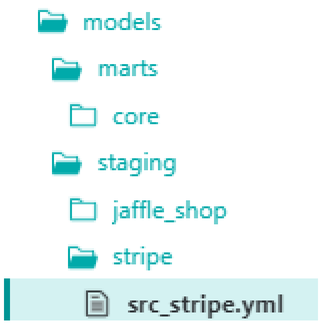
- ◆ Configure sources
    - I will configure the sources for the sql's of customers and orders in a file called scr_jaffle_shop.yml. Once I do that I will change the word raw to dbt-Tutorial for the database.


```
models
 marts
  core
   dim_customers.sql
   fct_orders.sql
 staging
  jaffle_shop
   src_jaffle_shop.yml
```

- ◆ Refactoring staging models
    - I will refactor stg_customers.sql, stg_orders.sql,refactor stg_payments.sql using the source function.

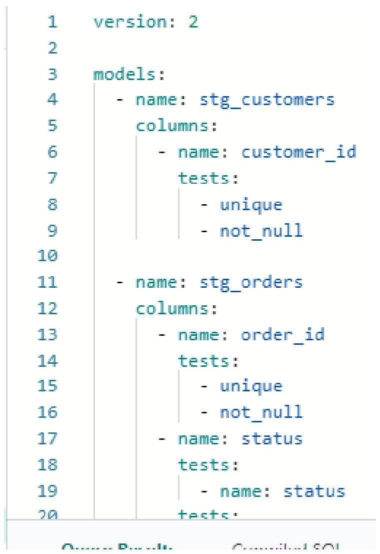

```
staging
 jaffle_shop
  src_jaffle_shop.yml
  stg_customers.sql
```

- **Source–Exemplar**
  - <u>Self-check src_stripe and stg_payments</u>
    - I created a file named scr_stripe.yml in the stripe folder

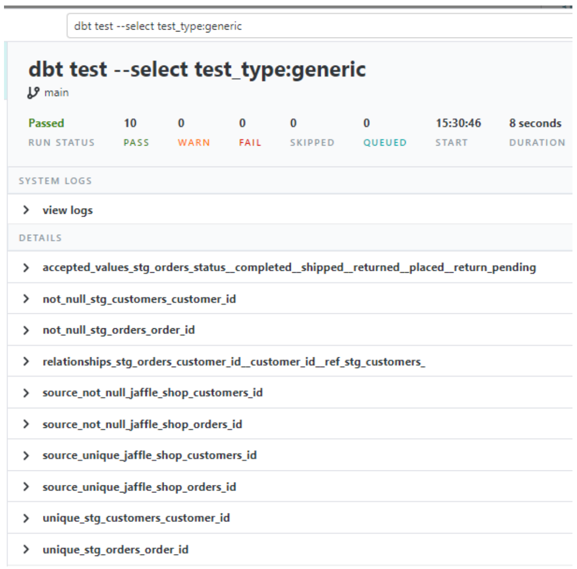


```
models
  marts
    core
  staging
    jaffle_shop
    stripe
      src_stripe.yml
```

- **Tests – Practice**
  - <u>Generic Tests</u>
    - I created a file called stg_jaffle_shop.yml for configuring my tests using the generic test. Where I added unique and not_null tests to the keys for each of my staging tables. The singular test then adds it to the stg_payments model. Finally I run my test like the generic test and the singular test.

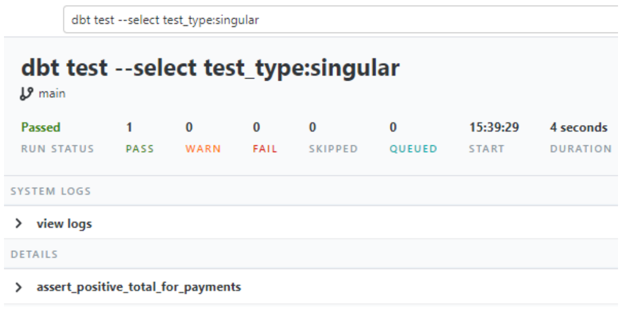


```yaml
version: 2

models:
  - name: stg_customers
    columns:
      - name: customer_id
        tests:
          - unique
          - not_null

  - name: stg_orders
    columns:
      - name: order_id
        tests:
          - unique
          - not_null
      - name: status
        tests:
          - name: status
        tests:
```



```
dbt test --select test_type:generic

dbt test --select test_type:generic
main

Passed        10     0      0      0         0        15:30:46   8 seconds
RUN STATUS    PASS   WARN   FAIL   SKIPPED   QUEUED   START      DURATION

SYSTEM LOGS
> view logs
DETAILS
> accepted_values_stg_orders_status__completed__shipped__returned__placed__return_pending
> not_null_stg_customers_customer_id
> not_null_stg_orders_order_id
> relationships_stg_orders_customer_id__customer_id__ref_stg_customers_
> source_not_null_jaffle_shop_customers_id
> source_not_null_jaffle_shop_orders_id
> source_unique_jaffle_shop_customers_id
> source_unique_jaffle_shop_orders_id
> unique_stg_customers_customer_id
> unique_stg_orders_order_id
```

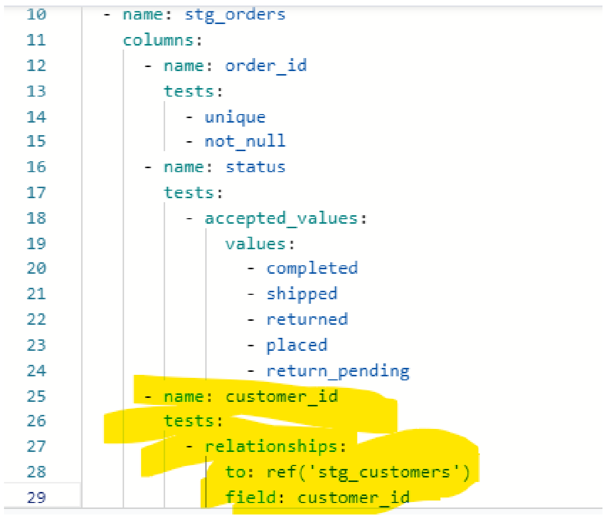
- <u>Singular Tests</u>
    - I added to the test folder by clicking the three dots and adding a file name called tests/assert_positive_value_for_total_amount.sql, where I'll be run on my stg_payments model.



```
dbt test --select test_type:singular

dbt test --select test_type:singular
main

Passed       1      0      0      0         0        15:39:29   4 seconds
RUN STATUS   PASS   WARN   FAIL   SKIPPED   QUEUED   START      DURATION

SYSTEM LOGS
> view logs
DETAILS
> assert_positive_total_for_payments
```

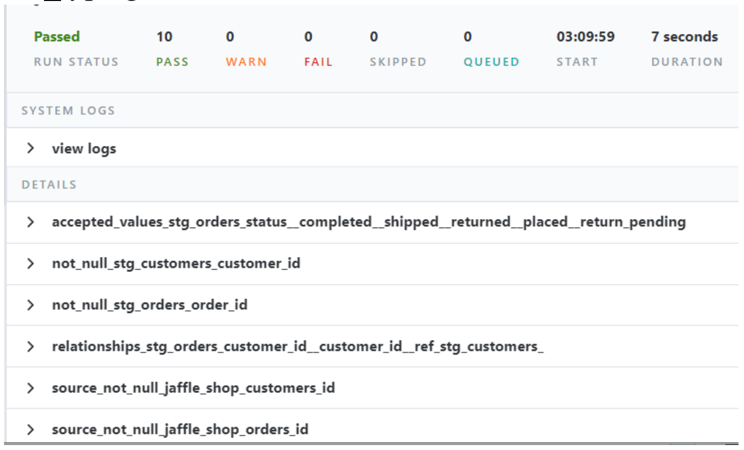
→ Test-Exemplar

- <u>Relationship test</u>
    - I added a relationships test to my stg_orders model for the customer_id in stg_customers. By inputting the code below



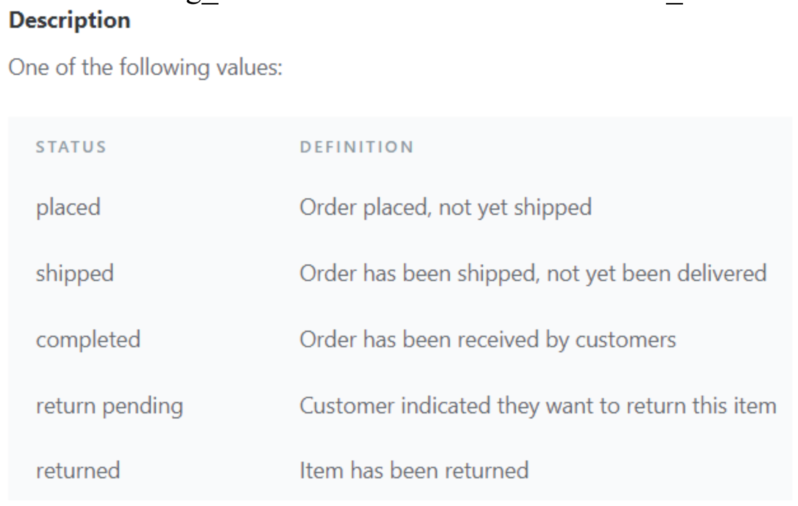
```
  - name: stg_orders
    columns:
      - name: order_id
        tests:
          - unique
          - not_null
      - name: status
        tests:
          - accepted_values:
              values:
                - completed
                - shipped
                - returned
                - placed
                - return_pending
      - name: customer_id
        tests:
          - relationships:
              to: ref('stg_customers')
              field: customer_id
```

Where I received 10 pass tests when running the dbt test –select test_type:generic.



```
Passed       10     0      0      0         0        03:09:59   7 seconds
RUN STATUS   PASS   WARN   FAIL   SKIPPED   QUEUED   START      DURATION

SYSTEM LOGS
> view logs
DETAILS
> accepted_values_stg_orders_status__completed__shipped__returned__placed__return_pending
> not_null_stg_customers_customer_id
> not_null_stg_orders_order_id
> relationships_stg_orders_customer_id__customer_id__ref_stg_customers_
> source_not_null_jaffle_shop_customers_id
> source_not_null_jaffle_shop_orders_id
```

- **Documentation – Practice**
    - Write documentation
        - I will next add documentation to the file models/staging/jaffle_shop/stg_jaffle_shop.yml. To do this I will run docs generate. I will also add a description for your stg_customers model and the column customer_id. Then add a different description for your stg_orders model and the column order_id.

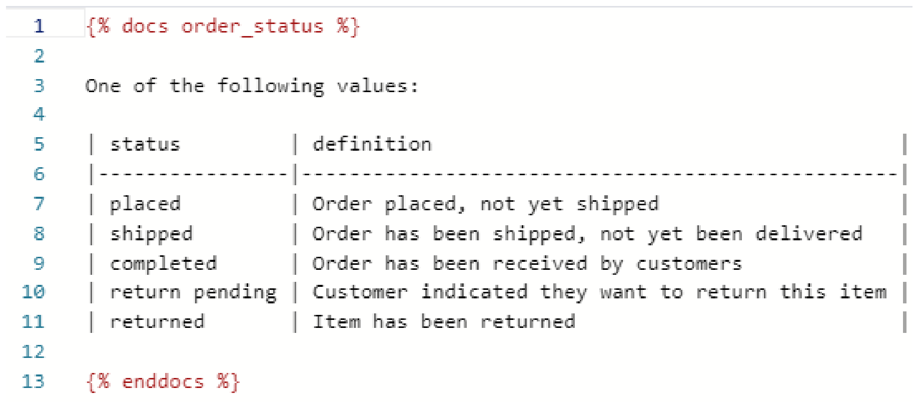


Description

One of the following values:

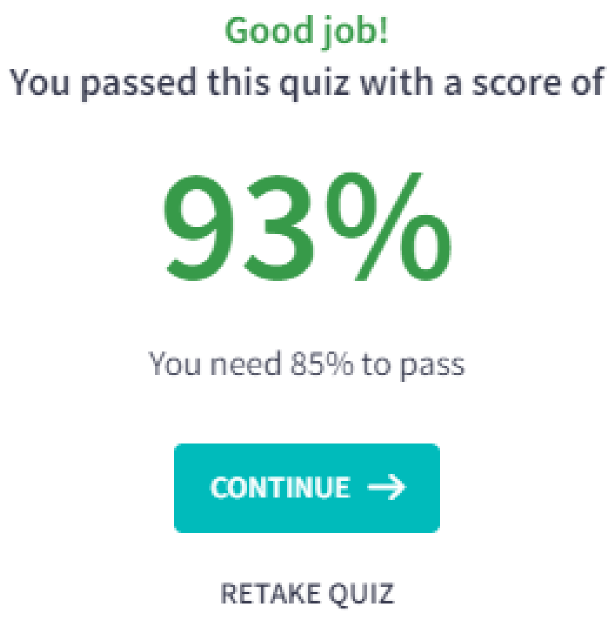
STATUS	DEFINITION
placed	Order placed, not yet shipped
shipped	Order has been shipped, not yet been delivered
completed	Order has been received by customers
return pending	Customer indicated they want to return this item
returned	Item has been returned

- **Create a reference to a doc block**
    - Create a doc block for your stg_orders model to document the status column. Where I will reference this doc block in the description of status in stg_orders.



```
{% docs order_status %}

One of the following values:

| status         | definition                                       |
|----------------|--------------------------------------------------|
| placed         | Order placed, not yet shipped                    |
| shipped        | Order has been shipped, not yet been delivered   |
| completed      | Order has been received by customers             |
| return pending | Customer indicated they want to return this item |
| returned       | Item has been returned                           |

{% enddocs %}
```

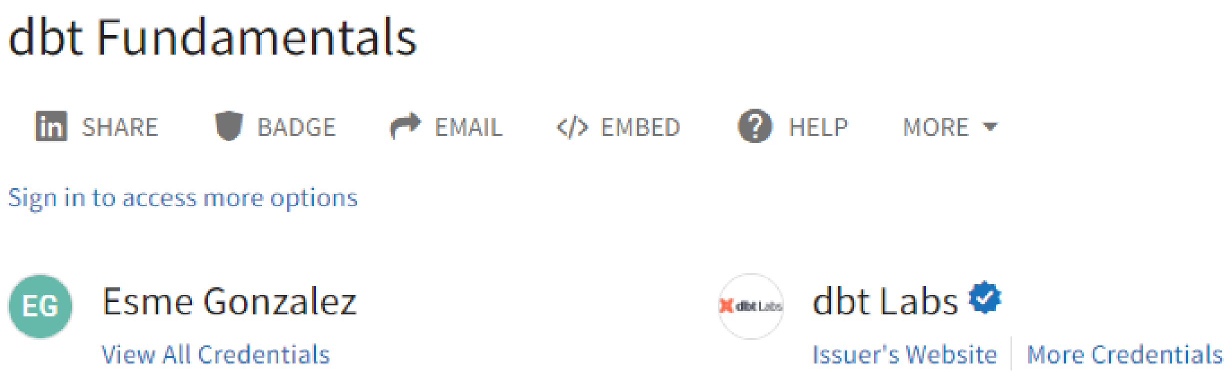
My results on the quiz is


Good job!

You passed this quiz with a score of

93%

You need 85% to pass


dbt Fundamentals

Sign in to access more options

Esme Gonzalez

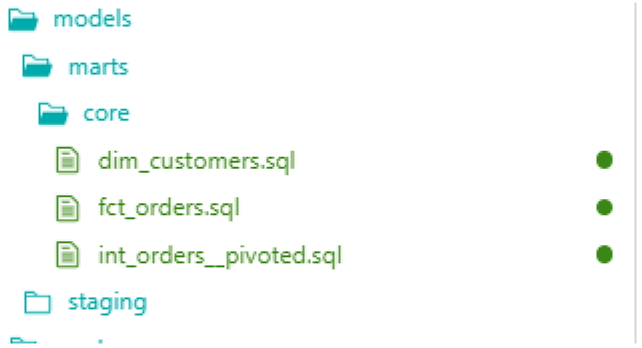
View All Credentials

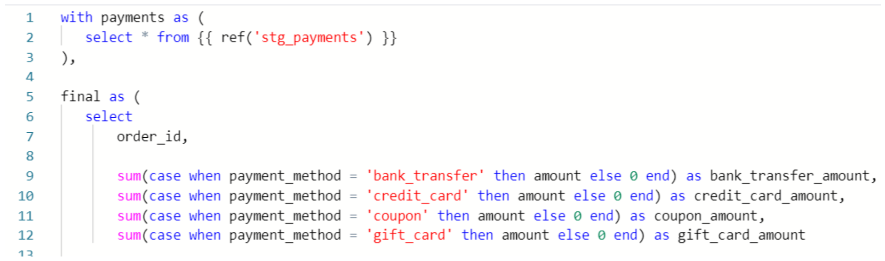
dbt Labs

Issuer's Website | More Credentials

Course 2!!

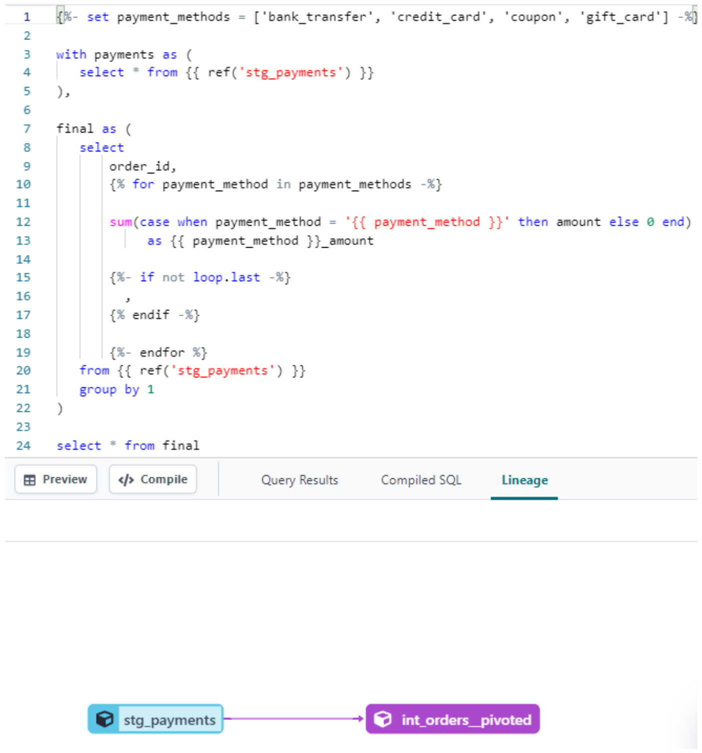
- ➔ **Jinja Primer – Practice**
  - ◆ I created a new file called int_orders_pivoted.sql. In my marts/core folder.





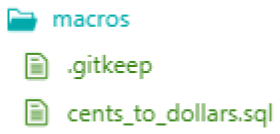
```sql
with payments as (
    select * from {{ ref('stg_payments') }}
),

final as (
    select
        order_id,

        sum(case when payment_method = 'bank_transfer' then amount else 0 end) as bank_transfer_amount,
        sum(case when payment_method = 'credit_card' then amount else 0 end) as credit_card_amount,
        sum(case when payment_method = 'coupon' then amount else 0 end) as coupon_amount,
        sum(case when payment_method = 'gift_card' then amount else 0 end) as gift_card_amount
```

- ➔ **Jinja Primer-Exemplar (do not include the Advanced Jinja + Macros grant_select_macro part)**
  - ◆ I would then enter the query below in my int_orders_pivoted.sql so that the order id can relate to the amount with a credit card, bank transfer, etc.

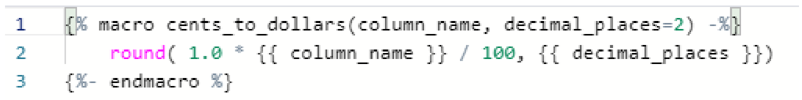


```sql
{%- set payment_methods = ['bank_transfer', 'credit_card', 'coupon', 'gift_card'] -%}

with payments as (
    select * from {{ ref('stg_payments') }}
),

final as (
    select
        order_id,
        {% for payment_method in payment_methods -%}

        sum(case when payment_method = '{{ payment_method }}' then amount else 0 end)
            as {{ payment_method }}_amount

        {%- if not loop.last -%}
         ,
        {% endif -%}

        {%- endfor %}
    from {{ ref('stg_payments') }}
    group by 1
)

select * from final
```

- **Macros – Practice**
  - Next, I will create a file in my macros folder called cents_to_dollars.sql. I will write the following code shown below.

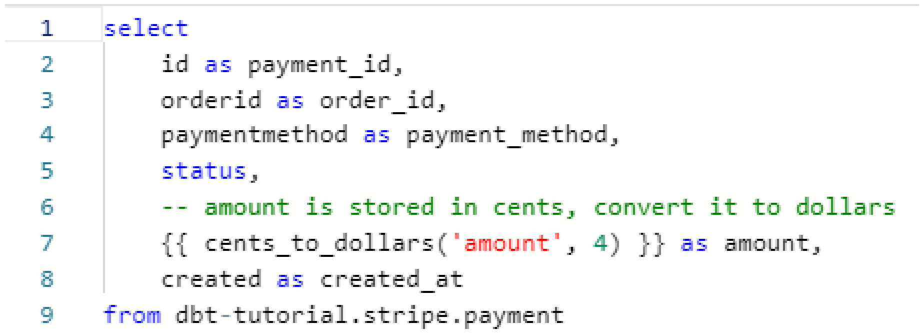


```
macros
  .gitkeep
  cents_to_dollars.sql
```



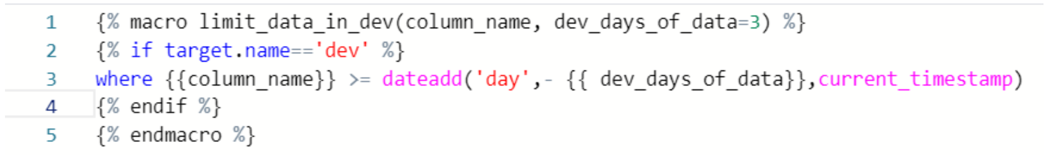
```
{% macro cents_to_dollars(column_name, decimal_places=2) -%}
    round( 1.0 * {{ column_name }} / 100, {{ decimal_places }})
{%- endmacro %}
```

  - Where I leverage my macro in my stg_payments.sql

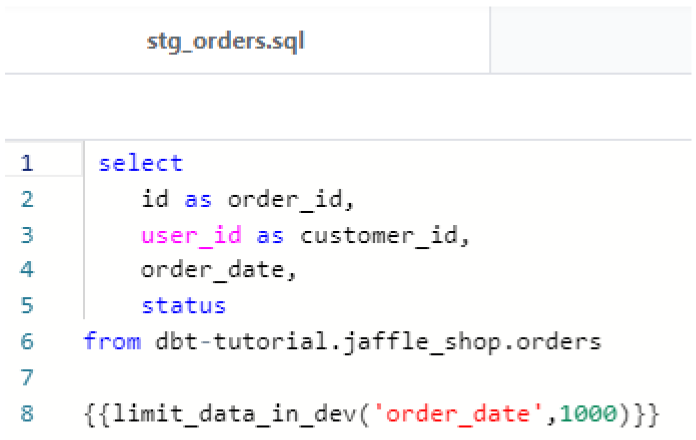


```
select
    id as payment_id,
    orderid as order_id,
    paymentmethod as payment_method,
    status,
    -- amount is stored in cents, convert it to dollars
    {{ cents_to_dollars('amount', 4) }} as amount,
    created as created_at
from dbt-tutorial.stripe.payment
```

- **Macros-Exemplar**
  - Next I will create limit_data_in_dev.sql and input what is below.



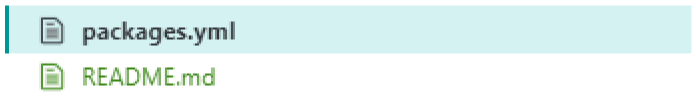
```
{% macro limit_data_in_dev(column_name, dev_days_of_data=3) %}
{% if target.name=='dev' %}
where {{column_name}} >= dateadd('day',- {{ dev_days_of_data}},current_timestamp)
{% endif %}
{% endmacro %}
```



stg_orders.sql

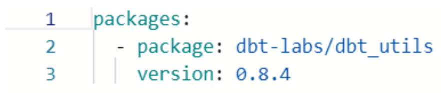
```
select
    id as order_id,
    user_id as customer_id,
    order_date,
    status
from dbt-tutorial.jaffle_shop.orders

{{limit_data_in_dev('order_date',1000)}}
```

- **Packages – Practice**
  - I created a new file called packages.yml.



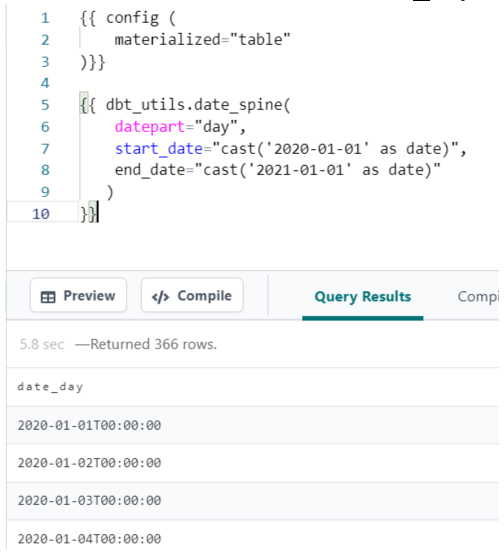
packages.yml
README.md

  - Where I input the code below to be able to download the new version of the packages. The link https://hub.getdbt.com/dbt-labs/dbt_utils/latest/ to find the code below. This model will be able to list every day in the year 2020.



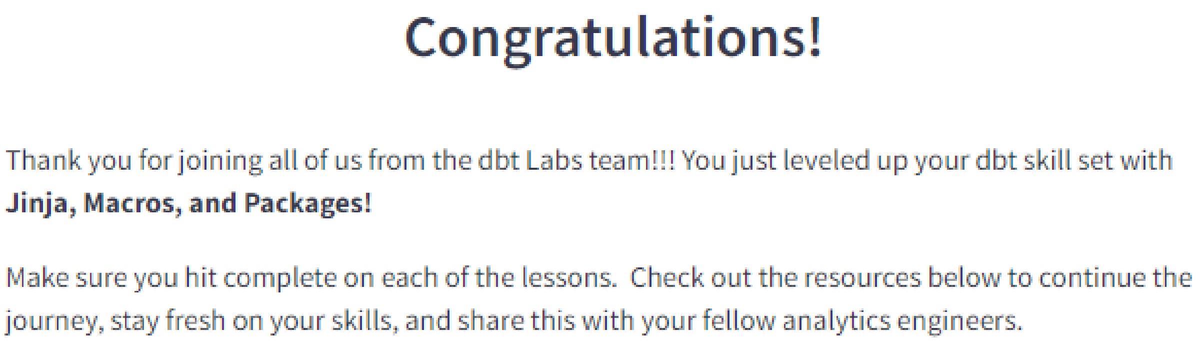
```
packages:
  - package: dbt-labs/dbt_utils
    version: 0.8.4
```

- **Packages -Exemplar**
  - I will then create a file called all_days.sql with the following code shown below.



```
{{ config (
    materialized="table"
)}}

{{ dbt_utils.date_spine(
    datepart="day",
    start_date="cast('2020-01-01' as date)",
    end_date="cast('2021-01-01' as date)"
   )
}}
```

```
5.8 sec  —Returned 366 rows.

date_day
2020-01-01T00:00:00
2020-01-02T00:00:00
2020-01-03T00:00:00
2020-01-04T00:00:00
```



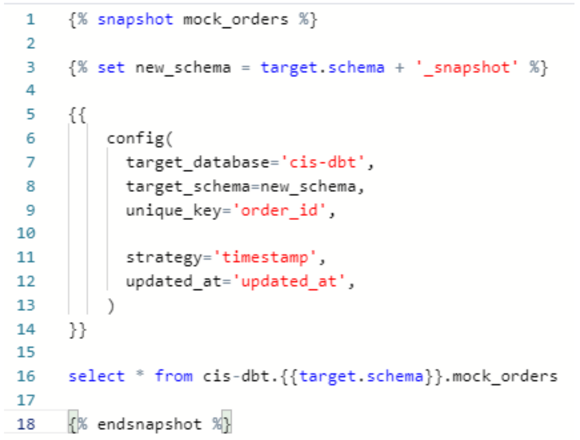
Congratulations!

Thank you for joining all of us from the dbt Labs team!!! You just leveled up your dbt skill set with **Jinja, Macros, and Packages!**

Make sure you hit complete on each of the lessons. Check out the resources below to continue the journey, stay fresh on your skills, and share this with your fellow analytics engineers.

Course 3!!

- **Materializations – Practice** (Skip the section on “Incremental Models”)
  - <u>Snapshots</u>
    - In my dbt I created a new snapshot in the folder snapshots with the filename mock_orders.sql with the following code below.



```
{% snapshot mock_orders %}

{% set new_schema = target.schema + '_snapshot' %}

{{
    config(
      target_database='cis-dbt',
      target_schema=new_schema,
      unique_key='order_id',

      strategy='timestamp',
      updated_at='updated_at',
    )
}}

select * from cis-dbt.{{target.schema}}.mock_orders

{% endsnapshot %}
```

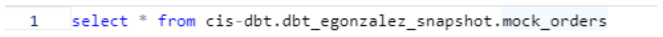
- Then I will put the code below in my Bigquery
  CREATE or REPLACE table cis-dbt.dbt_egonzalez.mock_orders

(
  order_id integer,
  status string,
  created_at date,
  updated_at date
);

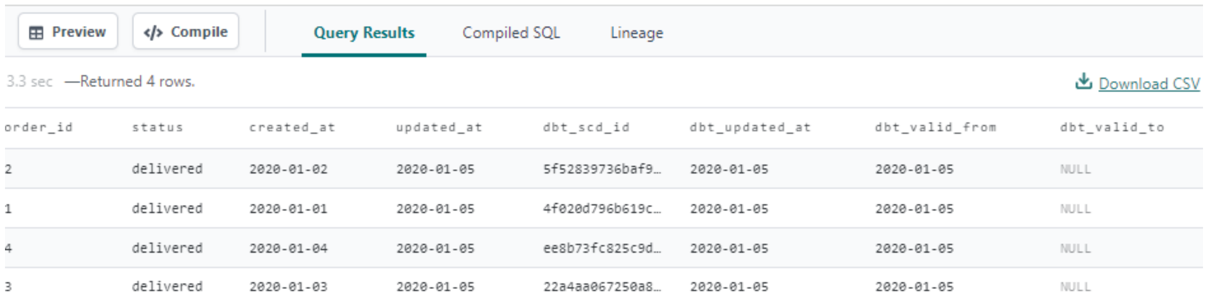
- To create my table then I will input the code below
  insert into cis-dbt.dbt_egonzalez.mock_orders (order_id, status, created_at,updated_at)

values (1, 'delivered', '2020-01-01', '2020-01-05'),
  (2, 'delivered', '2020-01-02', '2020-01-05'),
  (3, 'delivered', '2020-01-03', '2020-01-05'),
  (4, 'delivered', '2020-01-04', '2020-01-05');

- In my dbt I will then put the code below to produce my query results shown below.

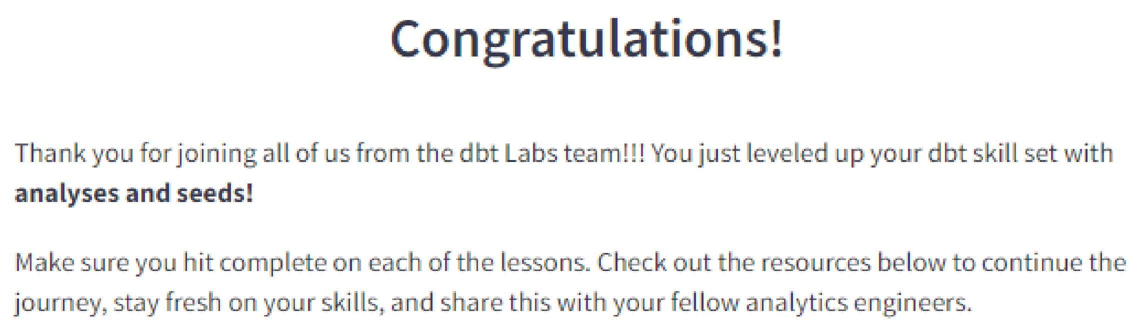


```
select * from cis-dbt.dbt_egonzalez_snapshot.mock_orders
```



3.3 sec —Returned 4 rows.

order_id	status	created_at	updated_at	dbt_scd_id	dbt_updated_at	dbt_valid_from	dbt_valid_to
2	delivered	2020-01-02	2020-01-05	5f52839736baf9…	2020-01-05	2020-01-05	NULL
1	delivered	2020-01-01	2020-01-05	4f020d796b619c…	2020-01-05	2020-01-05	NULL
4	delivered	2020-01-04	2020-01-05	ee8b73fc825c9d…	2020-01-05	2020-01-05	NULL
3	delivered	2020-01-03	2020-01-05	22a4aa067250a8…	2020-01-05	2020-01-05	NULL



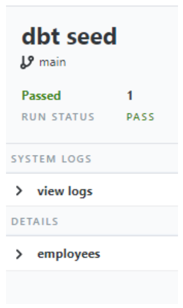
Congratulations!

Thank you for joining all of us from the dbt Labs team!!! You just leveled up your dbt skill set with **analyses and seeds!**

Make sure you hit complete on each of the lessons. Check out the resources below to continue the journey, stay fresh on your skills, and share this with your fellow analytics engineers.

Course 4!!

- **Analyses and Seeds – Practice**
  - I created a seed file in the seeds folder called employees.csv with the code seen below.



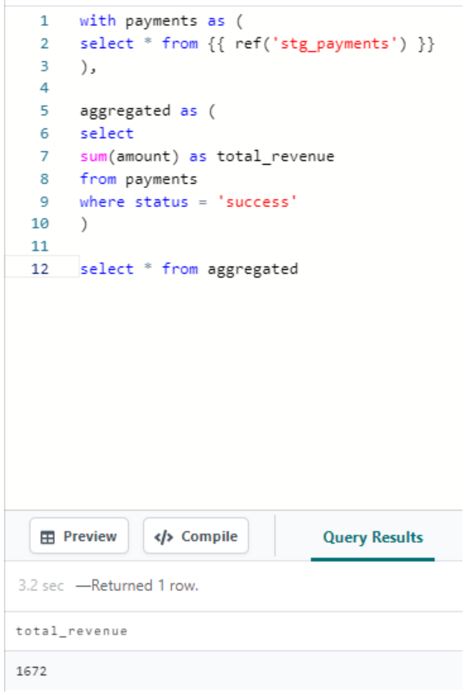
dbt seed

main

Passed 1

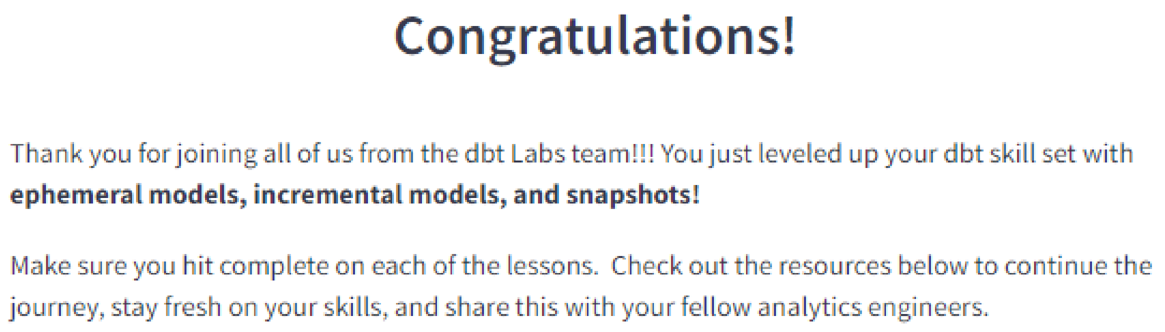
```
employee_id,email,customer_id
3425, mike@jaffleshop.com, 1
2354, sarah@jaffleshop.com, 6
2342, frank@jaffleshop.com, 8
1234, jennifer@jaffleshop.com, 9
```

- **Analyses and Seeds- Exemplar**
    - I create an analysis file in the analyses folder called total_revenue.sql that uses the stg_payments model and sums the amount of successful payments. Which gives me the query below



```
with payments as (
select * from {{ ref('stg_payments') }}
),

aggregated as (
select
sum(amount) as total_revenue
from payments
where status = 'success'
)

select * from aggregated
```

```
3.2 sec —Returned 1 row.
total_revenue
1672
```



Congratulations!

Thank you for joining all of us from the dbt Labs team!!! You just leveled up your dbt skill set with **ephemeral models, incremental models, and snapshots!**

Make sure you hit complete on each of the lessons. Check out the resources below to continue the journey, stay fresh on your skills, and share this with your fellow analytics engineers.

In conclusion, I would say this took me about 25 to 30 hours . I would say the most difficult part of this assignment was the fundamentals course part one. My reason is because I realize and learn alot from my mistakes. Where I miss something or didn't do something correctly. It definitely made me realize the importances of checking my work. Which made me back track. Overall, it was a wonderful experience!