

Part 2 – Data Build Tools Demo #1

1. Pertama buka aplikasi visual studio codenya, kemudian open folder dengan nama DBT-DEMO.
2. Lalu buat file docker-compose.yml sesuai dari repository yang ada.

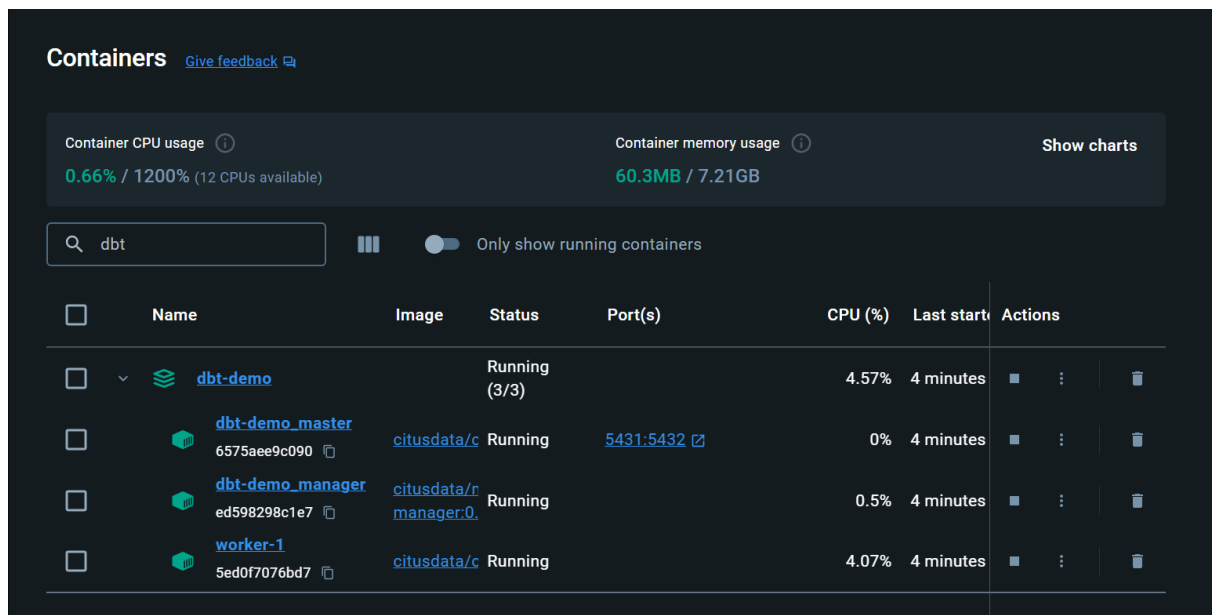
```
docker-compose.yml
1  version: '3'
2
3  services:
4
5      master:
6          container_name: "${COMPOSE_PROJECT_NAME:-citus}_master"
7          image: "citusdata/citus:12.0.0"
8          ports:
9              - "${COORDINATOR_EXTERNAL_PORT:-5431}:5432"
10         labels:
11             - "com.citusdata.role=Master"
12         environment: &AUTH
13             - POSTGRES_PASSWORD=pass
14             - POSTGRES_USER=postgres
15             - POSTGRES_DB=store
16             - PGUSER=postgres
17             - POSTGRES_HOST_AUTH_METHOD=trust
18         networks:
19             - postgres-network
20         volumes:
21             - ./citus-db-data:/var/lib/postgresql/data/
22             - ./init.sql:/docker-entrypoint-initdb.d/init.sql
23
24         worker:
25             image: "citusdata/citus:12.0.0"
26             labels:
27                 - "com.citusdata.role=Worker"
28             depends_on:
29                 - manager
30             environment: *AUTH
31             command: "/wait-for-manager.sh"
32             volumes:
33                 - ./citus-healthcheck:/healthcheck
```

Contoh Sebagian tampilan dari isi docker-compose.yml yang baru saja dibuat.

3. Jalankan docker dengan mengetik “docker compose up -d” pada terminal.

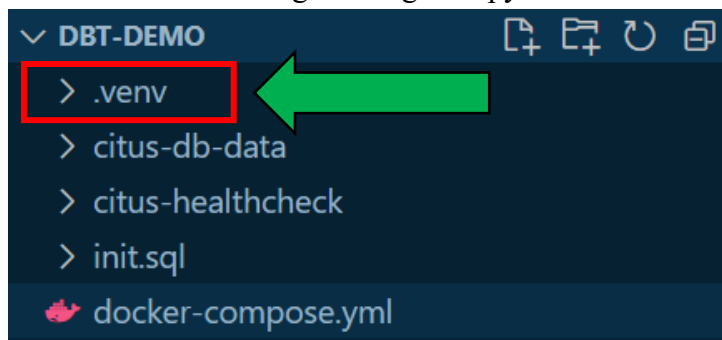
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS
tawhe@Ibnu_Emil MINGW64 ~/Documents/DBT-DEMO (main)
$ docker compose up -d
[+] Running 5/5
✓ Network dbt-demo_default          Created          0.1s
✓ Network dbt-demo_postgres-network Created          0.1s
✓ Container dbt-demo_master         Started          0.8s
✓ Container dbt-demo_manager        Started         1.1s
✓ Container dbt-demo-worker-1       Started         1.4s
```

Berikut merupakan tampilan perintah “docker compose up -d” sudah berjalan.



Container berhasil dibuat pada aplikasi docker.

4. Buat environment dengan mengetik “pyhton -m venv .venv”.



Environment sudah berhasil dibuat.

5. Aktifkan environment yang sudah dibuat sebelumnya dengan perintah “source .venv/Scripts/activate”.

```
tawhe@Ibnu_Emil MINGW64 ~/Documents/DBT-DEMO (main)
$ source .venv/Scripts/activate
(.venv)
tawhe@Ibnu_Emil MINGW64 ~/Documents/DBT-DEMO (main)
```

6. Install adapter dengan perintah “pip install dbt-postgres”.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

Using cached pydantic_core-2.20.1-cp312-none-win_amd64.whl.metadata (6.7 kB)
Downloading dbt_postgres-1.8.2-py3-none-any.whl (33 kB)
Downloading dbt_adapters-1.4.0-py3-none-any.whl (159 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 159.9/159.9 kB 737.2 kB/s eta 0:00:00
Downloading dbt_common-1.7.0-py3-none-any.whl (78 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 78.1/78.1 kB 1.1 MB/s eta 0:00:00
Downloading agate-1.9.1-py2.py3-none-any.whl (95 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 95.1/95.1 kB 1.1 MB/s eta 0:00:00
Downloading dbt_core-1.8.5-py3-none-any.whl (900 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 900.2/900.2 kB 280.6 kB/s eta 0:00:00
Using cached psycpg2_binary-2.9.9-cp312-cp312-win_amd64.whl (1.2 MB)
Downloading Babel-2.15.0-py3-none-any.whl (9.6 MB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.6/9.6 MB 293.8 kB/s eta 0:00:00
Using cached click-8.1.7-py3-none-any.whl (97 kB)
Using cached colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Downloading dbt_extractor-0.5.1-cp38-abi3-win_amd64.whl (283 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 283.5/283.5 kB 277.7 kB/s eta 0:00:00
Downloading dbt_semantic_interfaces-0.5.1-py3-none-any.whl (119 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 119.7/119.7 kB 319.0 kB/s eta 0:00:00
Downloading deepdiff-7.0.1-py3-none-any.whl (80 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 80.8/80.8 kB 205.2 kB/s eta 0:00:00
Downloading isodate-0.6.1-py2.py3-none-any.whl (41 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 41.7/41.7 kB 118.8 kB/s eta 0:00:00
```

Berikut adalah sebagian tampilan dari proses yang sedang berjalan dari perintah “pip install dbt-postgres”.

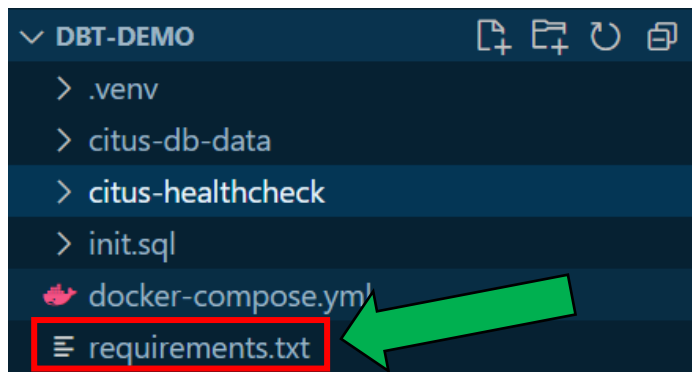
7. Lalu jalankan perintah untuk packages yang akan diinstall dengan perintah “pip freeze | grep dbt”.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

tawhe@Ibnu_Emil MINGW64 ~/Documents/DBT-DEMO (main)
● $ pip freeze | grep dbt
dbt-adapters==1.4.0
dbt-common==1.7.0
dbt-core==1.8.5
dbt-extractor==0.5.1
dbt-postgres==1.8.2
dbt-semantic-interfaces==0.5.1
(.venv)
tawhe@Ibnu_Emil MINGW64 ~/Documents/DBT-DEMO (main)
```

Berikut ini tampilan dari hasil perintah sebelumnya.

- Setelah itu jalankan perintah “`pip freeze | grep dbt >> requirements.txt`” untuk memasukkan packages ke dalam file `requirements.txt`.

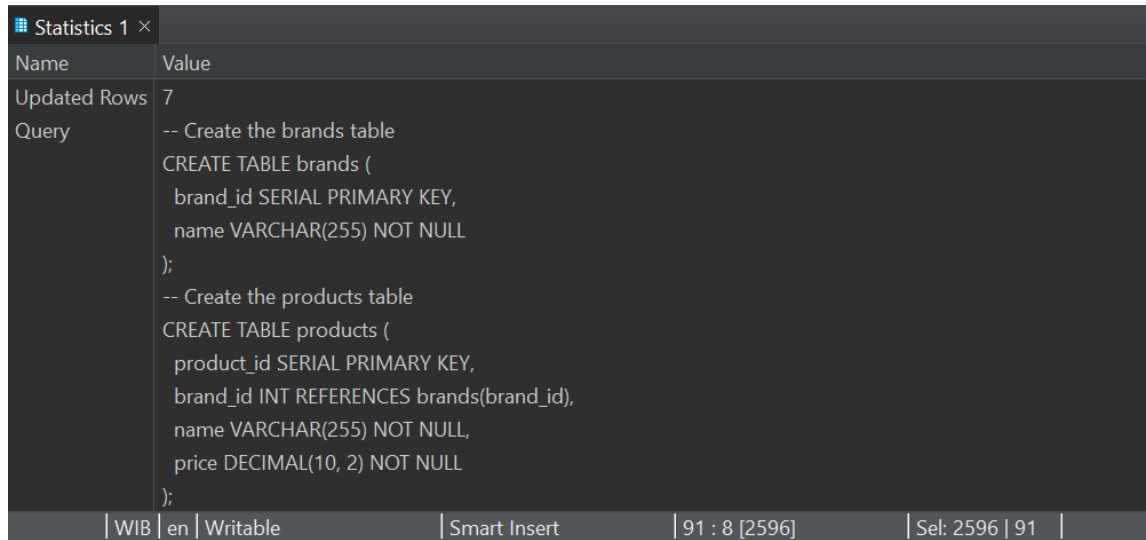


File “`requirements.txt`” sudah berhasil dibuat.

- Lalu jalankan file tersebut dengan perintah “`pip install -r requirements.txt`”
- Lakukan inisiasi kepada `my_project` dengan perintah “`dbt init my_project`” yang nanti akan menjadi folder utama dari dbt tersebut.
- Buat folder “`dbt-profiles`” lagi dengan perintah “`mkdir dbt-profiles`”.
- Di dalam folder “`dbt-profiles`” buat sebuah file dengan nama “`profiles.yml`” dengan perintah “`touch dbt-profiles/profiles.yml`”.
- Setelah itu jalankan perintah “`export DBT_PROFILES_DIR=$(pwd)/dbt-profiles`”.
- Kemudian isi file pada “`profiles.yml`” sesuai yang ada di repository.
Berikut ini adalah isi dari file “`profiles.yml`” yang sesuai dengan repository.

```
dbt-profiles > ! profiles.yml
1  my_project:
2    outputs:
3
4      dev:
5        type: postgres
6        threads: 1
7        host: localhost
8        port: 5431
9        user: postgres
10       pass: pass
11       dbname: store
12       schema: public
13
14     target: dev
```

15. Kemudian buka aplikasi DBeaver dan pilih connection dbname store dengan port 5431, lalu pilih Databases > store > Schemas > public > Tables. Pada Table buat query sesuai dengan repository dan jalankan query tersebut.



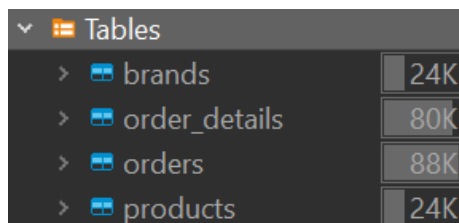
The screenshot shows the DBeaver SQL editor with a query to create two tables: 'brands' and 'products'. The 'brands' table has a primary key 'brand_id' and a 'name' column. The 'products' table has a primary key 'product_id', a foreign key 'brand_id' referencing 'brands', and 'name' and 'price' columns. The status bar at the bottom indicates 91 lines of code.

Name	Value
Updated Rows	7

```
-- Create the brands table
CREATE TABLE brands (
  brand_id SERIAL PRIMARY KEY,
  name VARCHAR(255) NOT NULL
);

-- Create the products table
CREATE TABLE products (
  product_id SERIAL PRIMARY KEY,
  brand_id INT REFERENCES brands(brand_id),
  name VARCHAR(255) NOT NULL,
  price DECIMAL(10, 2) NOT NULL
);
```

Berikut ini adalah hasil dari eksekusi query yang sudah dibuat sebelumnya, dan eksekusi berhasil.

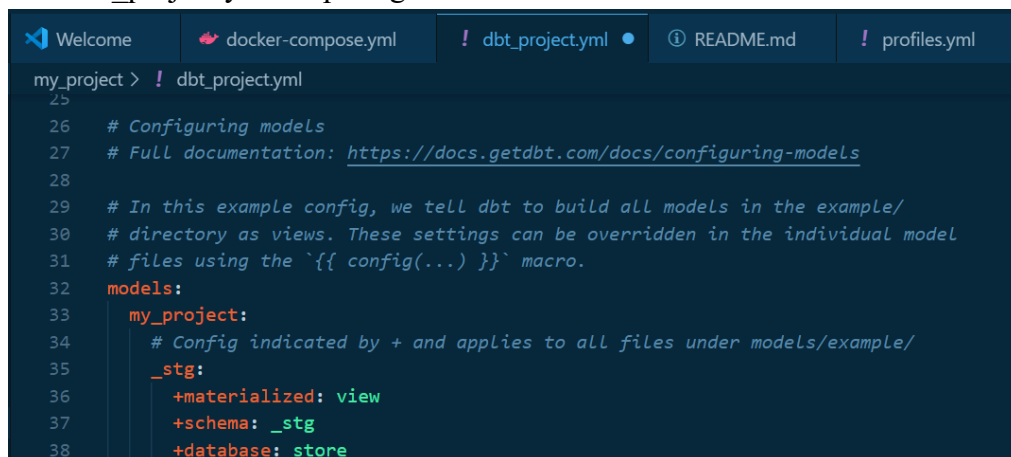


The screenshot shows the 'Tables' panel in DBeaver, listing four tables: 'brands' (24K), 'order_details' (80K), 'orders' (88K), and 'products' (24K).

Table Name	Size
brands	24K
order_details	80K
orders	88K
products	24K

Tabel sekarang sudah berhasil dibuat.

16. Set “dbt_project.yml” seperti gambar berikut ini :



The screenshot shows the 'dbt_project.yml' file in a code editor. The file contains configuration for the 'my_project' model, including settings for 'materialized', 'schema', and 'database'.

```
my_project > ! dbt_project.yml
25
26 # Configuring models
27 # Full documentation: https://docs.getdbt.com/docs/configuring-models
28
29 # In this example config, we tell dbt to build all models in the example/
30 # directory as views. These settings can be overridden in the individual model
31 # files using the `{{ config(...) }}` macro.
32 models:
33   my_project:
34     # Config indicated by + and applies to all files under models/example/
35     _stg:
36       +materialized: view
37       +schema: _stg
38       +database: store
```

17. Rubah nama “models/example” menjadi models/_stg agar sesuai dengan isi file dbt_project.yml.
18. Rubah nama “schema.yml” menjadi “raw_schema.yml” dan juga rubah isi dari file tersebut sesuai dengan repository.

19. Untuk membuat model, buat file “stg_schema.yml” pada folder “models/_stg” dan isilah sesuai repository seperti berikut ini :

```
my_project > models > _stg > ! stg_schema.yml
1  version: 2
2
3  models:
4    - name: daily_sales
5      description: "Aggregated sales metrics per day"
6      columns:
7        - name: order_date
8          description: "The date of the orders"
9          tests:
10           - not_null
11        - name: total_quantity
12          description: "Total quantity of products sold"
13          tests:
14           - not_null
15        - name: total_revenue
16          description: "Total revenue for the day"
17          tests:
18           - not_null
```

20. Pada file dengan nama “my_first_dbt_model.sql” dirubah menjadi “stg_brands.sql”, lalu buatlah query seperti berikut ini, karena ini format yang akan dipakai jika kita pertama kali mengambil dari raw :

```
my_project > models > _stg > stg_brands.sql
1  select
2    brand_id::int as brand_id
3    , name as brand_name
4  from {{ source('store', 'brands') }}
```

21. Pada file dengan nama “my_second_dbt_model.sql” dirubah menjadi “stg_products.sql”, lalu buatlah query seperti berikut ini :

```
my_project > models > _stg > stg_products.sql
1  select
2    product_id::int as product_id
3    , brand_id::int as brand_id
4    , name as product_name
5    , price as product_price
6  from {{ source('store', 'products') }}
```

22. Buat file dengan nama “stg_orders.sql”, lalu buatlah query seperti berikut ini :

```
my_project > models > _stg > stg_orders.sql
1  select
2      order_id::int as order_id
3      , order_date::timestamp as order_at
4      , customer_phone
5  from {{ source('store', 'orders') }}
```

23. Buat file dengan nama “stg_order_details.sql”, lalu buatlah query seperti berikut ini :

```
my_project > models > _stg > stg_order_details.sql
1  select
2      order_details_id::int as order_details_id
3      , order_id::int as order_id
4      , product_id::int as product_id
5      , quantity as order_qty
6      , price as unit_sales
7  from {{ source('store', 'order_details') }}
```

24. Buatlah query pada file “stg_schema.yml” seperti berikut ini :

```
my_project > models > _stg > ! stg_schema.yml
1  version: 2
2
3  models:
4      - name: stg_brands
5        description: "Brand staging"
6        columns:
7            - name: brand_id
8              tests:
9                  - not_null
10                 - unique
11
12      - name: stg_products
13        description: "Product staging"
14        columns:
15            - name: product_id
16              test:
17                  - not_null
18                  - unique
19
20      - name: brand_id
21        test:
22            - not_null
```

```
my_project > models > _stg > ! stg_schema.yml
3  models:
24      - name: stg_orders
25        description: "Order staging"
26        columns:
27            - name: order_id
28              test:
29                  - not_null
30                  - unique
31
32      - name: stg_order_details
33        description: "Order details staging"
34        columns:
35            - name: order_detail_id
36              test:
37                  - not_null
38                  - unique
```

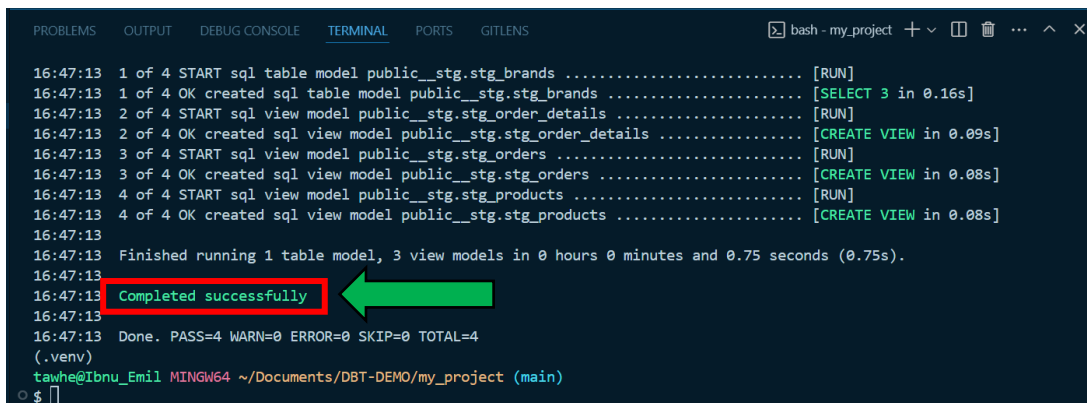
25. Buatlah query pada file “dbt_project.yml” seperti berikut ini :

```
my_project > ! dbt_project.yml

32  models:
33    my_project:
34      # Config indicated by + and applies to all files under models/example/
35      _stg:
36        +materialized: view
37        +schema: _stg
38        +database: store
39      -int:
40        +materialized: table
41        +schema: _int
42        +database: store
```

26. Lalu masuk ke folder “my_project” dengan perintah “cd my_project”.

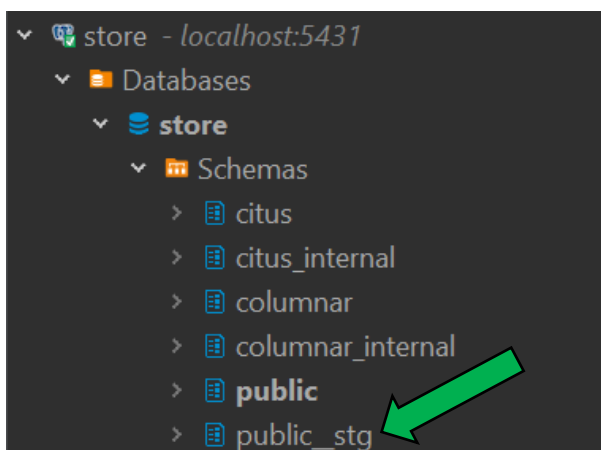
27. Kemudian jalankan dbt dengan perintah “dbt run”.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
bash - my_project + - - - - -

16:47:13 1 of 4 START sql table model public__stg.stg_brands ..... [RUN]
16:47:13 1 of 4 OK created sql table model public__stg.stg_brands ..... [SELECT 3 in 0.16s]
16:47:13 2 of 4 START sql view model public__stg.stg_order_details ..... [RUN]
16:47:13 2 of 4 OK created sql view model public__stg.stg_order_details ..... [CREATE VIEW in 0.09s]
16:47:13 3 of 4 START sql view model public__stg.stg_orders ..... [RUN]
16:47:13 3 of 4 OK created sql view model public__stg.stg_orders ..... [CREATE VIEW in 0.08s]
16:47:13 4 of 4 START sql view model public__stg.stg_products ..... [RUN]
16:47:13 4 of 4 OK created sql view model public__stg.stg_products ..... [CREATE VIEW in 0.08s]
16:47:13 Finished running 1 table model, 3 view models in 0 hours 0 minutes and 0.75 seconds (0.75s).
16:47:13 Completed successfully
16:47:13 Done. PASS=4 WARN=0 ERROR=0 SKIP=0 TOTAL=4
(.venv)
tawhe@Ibnu_Emil MINGW64 ~/Documents/DBT-DEMO/my_project (main)
$
```

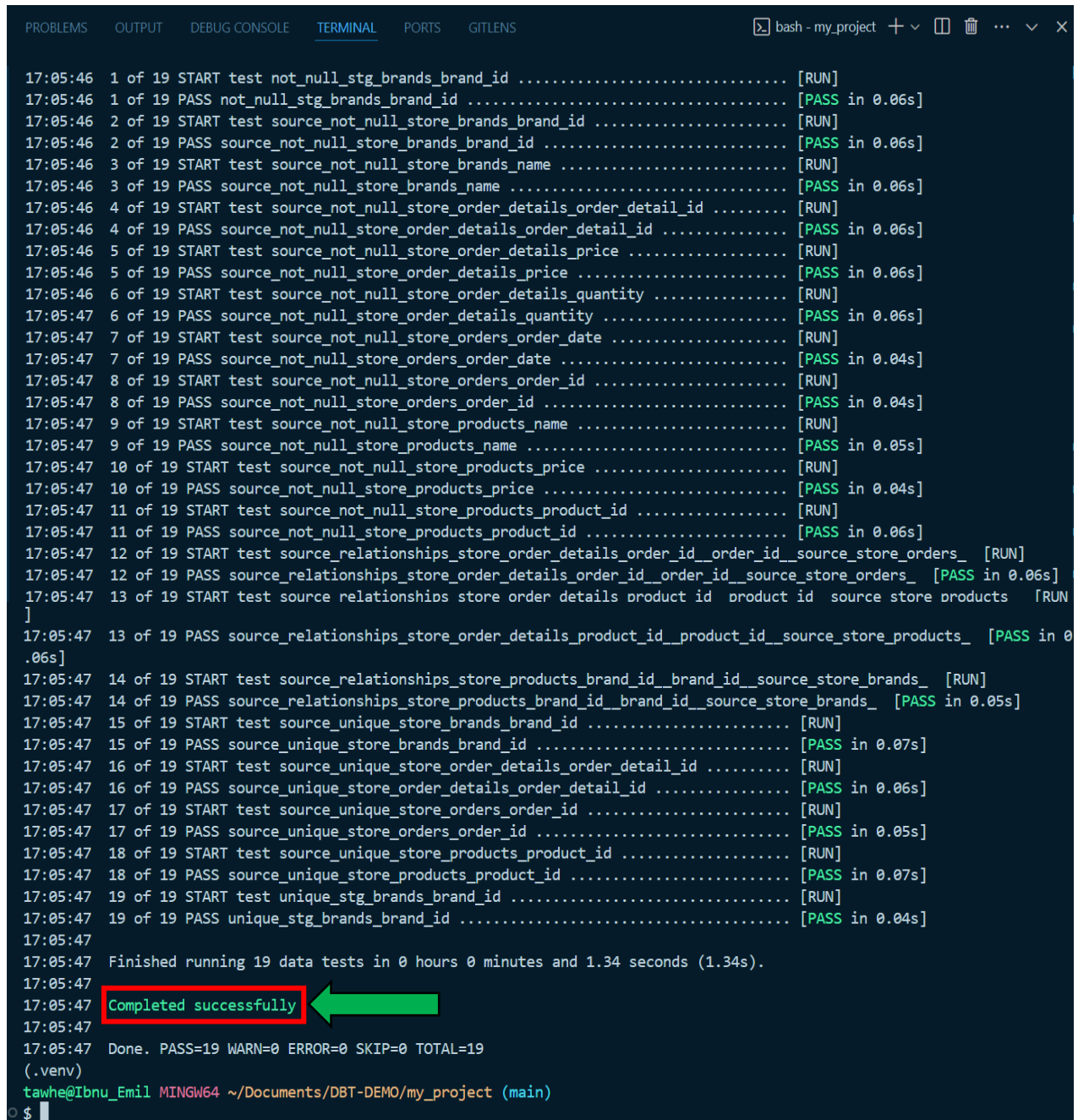
DBT berjalan dengan sesuai harapan, semua sukses, ditandai dengan tulisan **PASS=4**, **WARN=0**, **ERROR=0**, **SKIP=0**, **TOTAL=0** dan juga tulisan “**Completed successfully**”



```
store - localhost:5431
├── Databases
│   └── store
│       ├── Schemas
│       │   ├── citus
│       │   ├── citus_internal
│       │   ├── columnar
│       │   ├── columnar_internal
│       │   ├── public
│       │   └── public_stg
│       └── Tables
```

Pada aplikasi DBeaver juga dapat dibuktikan bahwa skema public__stg berhasil dibuat.

28. Langkah terakhir adalah dengan menguji apakah DBT yang sudah dibuat, sesuai dengan yang diharapkan. Untuk itu lakukan perintah “dbt test” pada terminal.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
bash - my_project

17:05:46 1 of 19 START test not_null_stg_brands_brand_id ..... [RUN]
17:05:46 1 of 19 PASS not_null_stg_brands_brand_id ..... [PASS in 0.06s]
17:05:46 2 of 19 START test source_not_null_store_brands_brand_id ..... [RUN]
17:05:46 2 of 19 PASS source_not_null_store_brands_brand_id ..... [PASS in 0.06s]
17:05:46 3 of 19 START test source_not_null_store_brands_name ..... [RUN]
17:05:46 3 of 19 PASS source_not_null_store_brands_name ..... [PASS in 0.06s]
17:05:46 4 of 19 START test source_not_null_store_order_details_order_detail_id ..... [RUN]
17:05:46 4 of 19 PASS source_not_null_store_order_details_order_detail_id ..... [PASS in 0.06s]
17:05:46 5 of 19 START test source_not_null_store_order_details_price ..... [RUN]
17:05:46 5 of 19 PASS source_not_null_store_order_details_price ..... [PASS in 0.06s]
17:05:46 6 of 19 START test source_not_null_store_order_details_quantity ..... [RUN]
17:05:47 6 of 19 PASS source_not_null_store_order_details_quantity ..... [PASS in 0.06s]
17:05:47 7 of 19 START test source_not_null_store_orders_order_date ..... [RUN]
17:05:47 7 of 19 PASS source_not_null_store_orders_order_date ..... [PASS in 0.04s]
17:05:47 8 of 19 START test source_not_null_store_orders_order_id ..... [RUN]
17:05:47 8 of 19 PASS source_not_null_store_orders_order_id ..... [PASS in 0.04s]
17:05:47 9 of 19 START test source_not_null_store_products_name ..... [RUN]
17:05:47 9 of 19 PASS source_not_null_store_products_name ..... [PASS in 0.05s]
17:05:47 10 of 19 START test source_not_null_store_products_price ..... [RUN]
17:05:47 10 of 19 PASS source_not_null_store_products_price ..... [PASS in 0.04s]
17:05:47 11 of 19 START test source_not_null_store_products_product_id ..... [RUN]
17:05:47 11 of 19 PASS source_not_null_store_products_product_id ..... [PASS in 0.06s]
17:05:47 12 of 19 START test source_relationships_store_order_details_order_id_order_id_source_store_orders_ [RUN]
17:05:47 12 of 19 PASS source_relationships_store_order_details_order_id_order_id_source_store_orders_ [PASS in 0.06s]
17:05:47 13 of 19 START test source_relationships_store_order_details_product_id_product_id_source_store_products_ [RUN]
17:05:47 13 of 19 PASS source_relationships_store_order_details_product_id_product_id_source_store_products_ [PASS in 0.06s]
17:05:47 14 of 19 START test source_relationships_store_products_brand_id_brand_id_source_store_brands_ [RUN]
17:05:47 14 of 19 PASS source_relationships_store_products_brand_id_brand_id_source_store_brands_ [PASS in 0.05s]
17:05:47 15 of 19 START test source_unique_store_brands_brand_id ..... [RUN]
17:05:47 15 of 19 PASS source_unique_store_brands_brand_id ..... [PASS in 0.07s]
17:05:47 16 of 19 START test source_unique_store_order_details_order_detail_id ..... [RUN]
17:05:47 16 of 19 PASS source_unique_store_order_details_order_detail_id ..... [PASS in 0.06s]
17:05:47 17 of 19 START test source_unique_store_orders_order_id ..... [RUN]
17:05:47 17 of 19 PASS source_unique_store_orders_order_id ..... [PASS in 0.05s]
17:05:47 18 of 19 START test source_unique_store_products_product_id ..... [RUN]
17:05:47 18 of 19 PASS source_unique_store_products_product_id ..... [PASS in 0.07s]
17:05:47 19 of 19 START test unique_stg_brands_brand_id ..... [RUN]
17:05:47 19 of 19 PASS unique_stg_brands_brand_id ..... [PASS in 0.04s]
17:05:47
17:05:47 Finished running 19 data tests in 0 hours 0 minutes and 1.34 seconds (1.34s).
17:05:47 Completed successfully
17:05:47
17:05:47 Done. PASS=19 WARN=0 ERROR=0 SKIP=0 TOTAL=19
(.venv)
tawhe@Ibnu_Emil MINGW64 ~/Documents/DBT-DEMO/my_project (main)
$
```

Dari hasil diatas dapat disimpulkan bahwa hasil dari perintah “dbt test” berhasil sesuai dengan yang diinginkan, ditandai dengan hasil **PASS=19, WARN=0, ERROR=0, SKIP=0, TOTAL=19** dan juga tulisan “Completed successfully”