DATA ENGINEER BATCH 4
Mentee: Yovina Silvia
Mentor: Bilal Benefit

TASK
DATA BUILD TOOL (DBT)

1. Jelaskan Apa Itu DBT

DBT (Data Build Tool) adalah alat yang digunakan untuk transformasi data di dalam gudang data modern. DBT memungkinkan data analis dan data engineer untuk mengubah data dengan menulis SQL sederhana yang dapat dipelihara. DBT digunakan untuk mengelola transformasi data yang kompleks dengan cara yang lebih terstruktur dan terorganisir. Alat ini memudahkan untuk menulis, menguji, dan memelihara transformasi data di dalam repositori versi kontrol seperti Git.

2. Apa Keuntungan Menggunakan DBT

Keuntungan menggunakan DBT meliputi:

a. **Versi Kontrol**: Semua perubahan pada transformasi data dapat dikelola dengan Git, memungkinkan pelacakan perubahan dan kolaborasi tim.
b. **Modularisasi dan Reusabilitas**: DBT mempromosikan penulisan SQL yang modular dan dapat digunakan kembali, yang mengurangi redundansi dan meningkatkan keterbacaan kode.
c. **Pemeriksaan Kualitas Data**: DBT mendukung pengujian data secara otomatis, memastikan bahwa transformasi data menghasilkan output yang benar dan dapat diandalkan.
d. **Dokumentasi Otomatis**: DBT menghasilkan dokumentasi secara otomatis untuk setiap transformasi, sehingga memudahkan pemahaman dan pemeliharaan pipeline data.
e. **Peningkatan Kecepatan Pengembangan**: Dengan fitur otomatisasi dan alur kerja yang lebih baik, DBT mempercepat proses pengembangan transformasi data.

3. Jelaskan Dependency Tree dan Versioning pada DBT

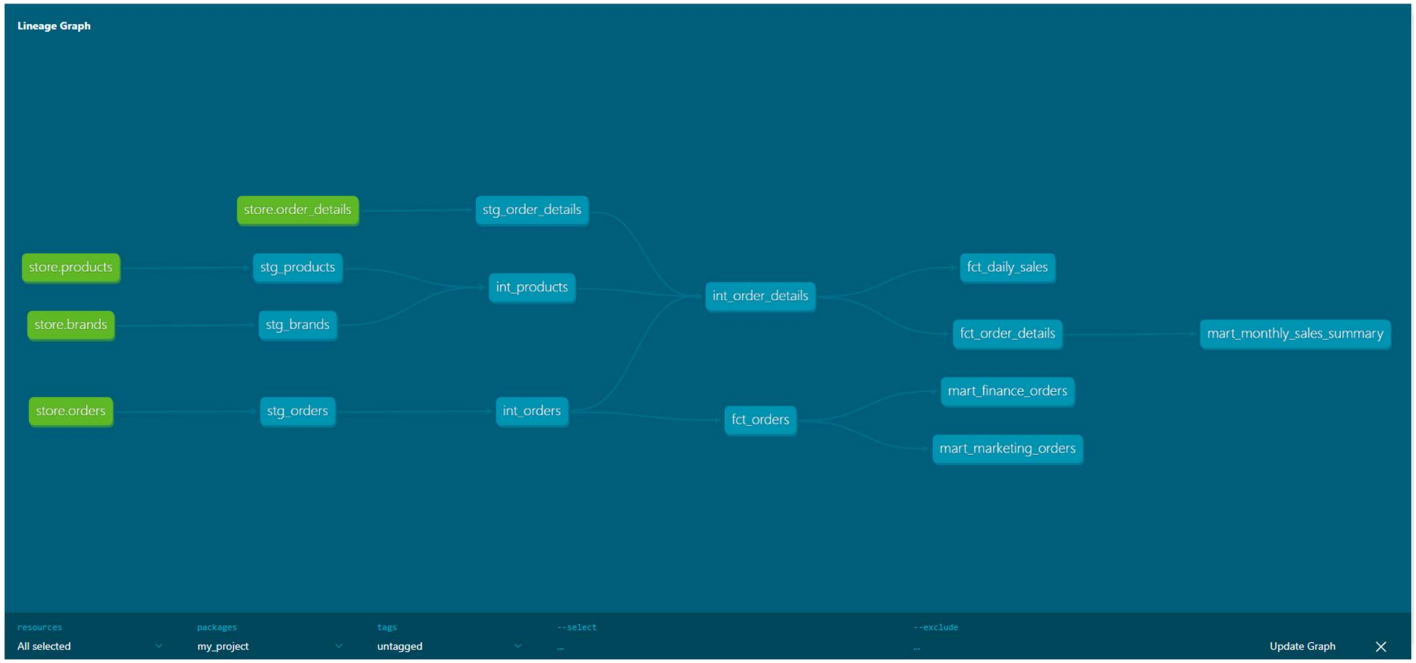
- **Dependency Tree**: Dalam DBT, dependency tree adalah struktur yang menunjukkan bagaimana model-model data saling bergantung satu sama lain. Setiap model (representasi SQL dari data yang ditransformasikan) dapat bergantung pada model lain. Dependency tree ini

membantu dalam memahami urutan eksekusi dan memastikan bahwa transformasi data dilakukan dengan benar. DBT secara otomatis menghasilkan dan mengelola dependency tree ini berdasarkan definisi model dan hubungan antar model yang dibuat oleh pengguna.

- **Versioning**: DBT memungkinkan pengelolaan versi transformasi data melalui integrasi dengan sistem kontrol versi seperti Git. Setiap perubahan pada transformasi data dapat dicatat dan dilacak dalam repositori Git, yang memungkinkan rollback, audit trail, dan kolaborasi yang lebih baik. Versioning juga memungkinkan tim untuk bekerja pada fitur baru atau perbaikan bug di cabang yang terpisah tanpa mengganggu alur kerja utama, dan kemudian menggabungkan perubahan tersebut kembali ke cabang utama setelah diuji dan diverifikasi.

Dependency Tree

Dependency tree pada DBT menunjukkan bagaimana model-model data saling bergantung satu sama lain. Setiap model data adalah representasi SQL dari tabel atau view yang dihasilkan dari transformasi data. Struktur ini membantu memahami urutan eksekusi dan memastikan bahwa transformasi data dilakukan dengan benar. **Contoh Dependency Tree:**



1. **Sumber Data Mentah**:
   - `store.products`, `store.brands`, `store.orders`, dan `store.order_details` adalah tabel-tabel mentah yang menjadi dasar dari semua transformasi berikutnya.
2. **Staging Models**:
   - **`stg_products`**: Mengambil data dari `store.products`.
   - **`stg_brands`**: Mengambil data dari `store.brands`.
   - **`stg_orders`**: Mengambil data dari `store.orders`.
   - **`stg_order_details`**: Mengambil data dari `store.order_details`.
3. **Intermediate Models**:
   - **`int_products`**: Mengambil data dari `stg_products` dan `stg_brands`, mungkin menggabungkan atau membersihkan data untuk digunakan lebih lanjut.
   - **`int_orders`**: Mengambil data dari `stg_orders`.
   - **`int_order_details`**: Mengambil data dari `stg_order_details` dan `int_products`, menggabungkan detail order dengan informasi produk.
4. **Fact Models**:
   - **`fct_orders`**: Mengambil data dari `int_orders`.
   - **`fct_order_details`**: Mengambil data dari `int_order_details`.
   - **`fct_daily_sales`**: Mengambil data dari `fct_order_details`, mungkin untuk menghitung penjualan harian.
5. **Mart Models**:
   - **`mart_finance_orders`**: Mengambil data dari `fct_orders`, mungkin untuk kebutuhan pelaporan keuangan.
   - **`mart_marketing_orders`**: Mengambil data dari `fct_orders`, mungkin untuk kebutuhan analisis pemasaran.
   - **`mart_monthly_sales_summary`**: Mengambil data dari `fct_daily_sales` dan `fct_order_details`, mungkin untuk ringkasan penjualan bulanan.

Urutan Eksekusi dan Alur Data:

1. Data mentah diambil dari tabel sumber (`store.products`, `store.brands`, `store.orders`, `store.order_details`).
2. Data ini diolah ke dalam staging models (`stg_`), yang bertugas membersihkan dan mempersiapkan data untuk transformasi lebih lanjut.
3. Intermediate models (`int_`) menggabungkan dan memproses data dari staging models untuk membuat data yang lebih siap digunakan.
4. Fact models (`fct_`) menghasilkan tabel fakta yang berisi data agregat dan metrik yang digunakan untuk analisis bisnis.

5. Mart models (`mart_`) digunakan untuk membuat laporan dan analisis spesifik berdasarkan kebutuhan departemen atau fungsi bisnis tertentu (seperti keuangan, pemasaran, dan ringkasan penjualan).

Versioning

Versioning pada DBT dikelola melalui integrasi dengan sistem kontrol versi seperti Git. Ini memungkinkan pengguna untuk melacak perubahan, berkolaborasi, dan menjaga riwayat perubahan pada transformasi data.

Contoh Alur Kerja Versioning:

1. **Inisialisasi Repositori Git**: Buat repositori Git untuk proyek DBT.

   git init
   git add .
   git commit -m "Initial commit"

2. **Membuat Cabang Fitur**: Buat cabang baru untuk fitur atau perbaikan bug.
   bash
   Copy code
   git checkout -b new_feature_branch

3. **Mengembangkan dan Menguji Perubahan**: Lakukan perubahan pada model dan pengujian di cabang fitur.

   git add .
   git commit -m "Add new feature"

4. **Merge ke Cabang Utama**: Setelah perubahan diuji dan diverifikasi, gabungkan cabang fitur kembali ke cabang utama.

   git checkout main
   git merge new_feature_branch

5. **Push ke Repositori Remote**: Push perubahan ke repositori remote untuk kolaborasi dengan tim.
   bash
   Copy code
   git push origin main

Dengan dependency tree, kita bisa melihat hubungan antara model-model data dan urutan eksekusi mereka. Dengan versioning, kita bisa melacak perubahan dan bekerja secara kolaboratif dalam tim, memastikan setiap perubahan tercatat dan dapat diatur kembali jika diperlukan. Versioning adalah aspek penting dalam pengelolaan proyek DBT. Dengan menggunakan Git, kita dapat mengelola perubahan, bekerja secara kolaboratif, dan memastikan bahwa transformasi data terorganisir dan terdokumentasi dengan baik. Dependency tree membantu memahami urutan eksekusi model, sementara versioning dengan Git memastikan semua perubahan tercatat dan dapat dilacak.

DBT-DEMO

1. Instalasi dan Persiapan

Pastikan telah menginstal `dbt`, `Docker`, dan `Python` di sistem.

2. Membuat dan Mengaktifkan Virtual Environment (venv)

Mengatur proyek DBT dengan virtual environment untuk pengembangan lokal

- Membuat Virtual Environment:
  `python3 -m venv .env`
- Mengaktifkan Virtual Environment:
  `source .venv/Scripts/activate`

3. Menginstal DBT: Setelah mengaktifkan virtual environment, instal DBT:

```
pip install dbt-postgres
```

4. Membuat Struktur Direktori DBT

Buat struktur direktori menggunakan perintah berikut di terminal:

```
mkdir dbt-new
cd dbt-new

mkdir -p my_project/models/store/_stg
mkdir -p my_project/models/store/_int
mkdir -p my_project/models/store/_fct
mkdir -p my_project/models/store/_mart
mkdir my_project/macros
touch my_project/dbt_project.yml
touch my_project/packages.yml
touch my_project/package-lock.yml
touch my_project/sources.yml
```

```
mkdir dbt-profiles
touch dbt-profiles/profiles.yml

touch docker.compose.yml
touch requirements.txt
touch init.sql
```

5. Menjalankan Docker

Jalankan docker compose untuk menjalankan proyek dalam container yang terisolasi:

```
docker-compose up -d
```

6. Menambahkan Konten File

a. Layer Staging pada DBT

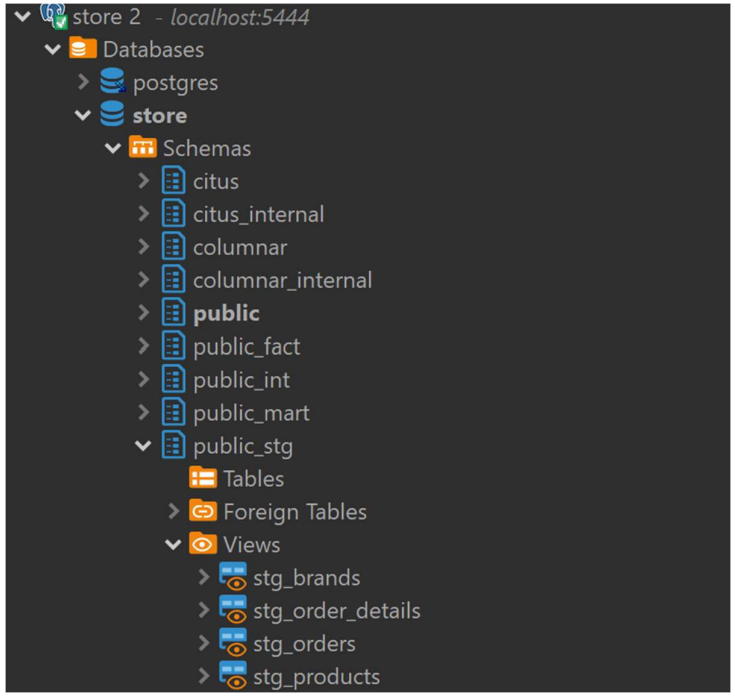
Layer staging pada DBT adalah salah satu lapisan penting dalam proses transformasi data. Tujuan utama dari layer ini adalah untuk membersihkan, memvalidasi, dan mempersiapkan data mentah dari berbagai sumber sebelum data tersebut digunakan dalam transformasi lebih lanjut atau dimuat ke dalam model yang lebih kompleks. Layer staging bertindak sebagai fondasi untuk model-model yang lebih lanjut dan memastikan bahwa data dalam keadaan yang bersih dan terstruktur dengan baik.

Fungsi Utama Layer Staging

1. **Membersihkan Data**: Menghapus data yang tidak valid, duplikat, atau data yang tidak lengkap.
2. **Mengubah Format Data**: Mengubah tipe data atau format data agar sesuai dengan standar yang diperlukan untuk analisis lebih lanjut.
3. **Validasi Data**: Memastikan bahwa data memenuhi aturan bisnis atau kriteria tertentu sebelum diproses lebih lanjut.
4. **Menyiapkan Data untuk Transformasi**: Menyusun data sehingga siap untuk digunakan dalam model intermediate atau fact.

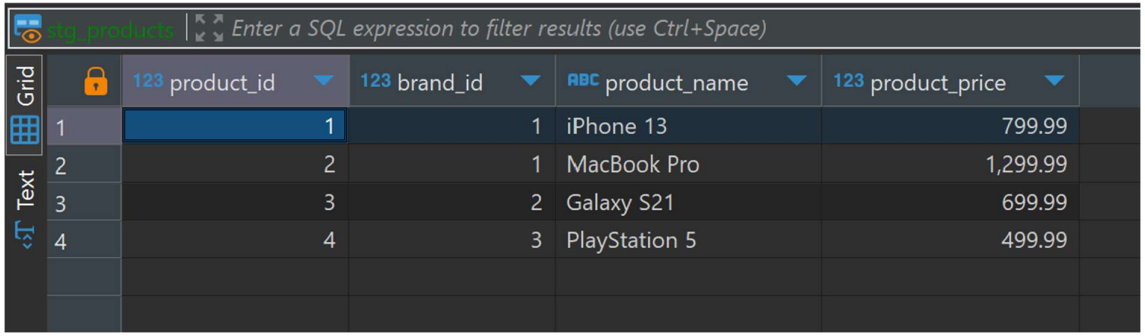
`my_project/models/store/_stg/`

Staging layer ini digunakan untuk membersihkan dan menyiapkan data mentah dari sumber data yang ada untuk tahap selanjutnya.



1. **stg_products.sql**

```
select
    product_id::int as product_id
    , brand_id::int as brand_id
    , name as product_name
    , price::float as product_price
from {{ source('store', 'products') }}
```



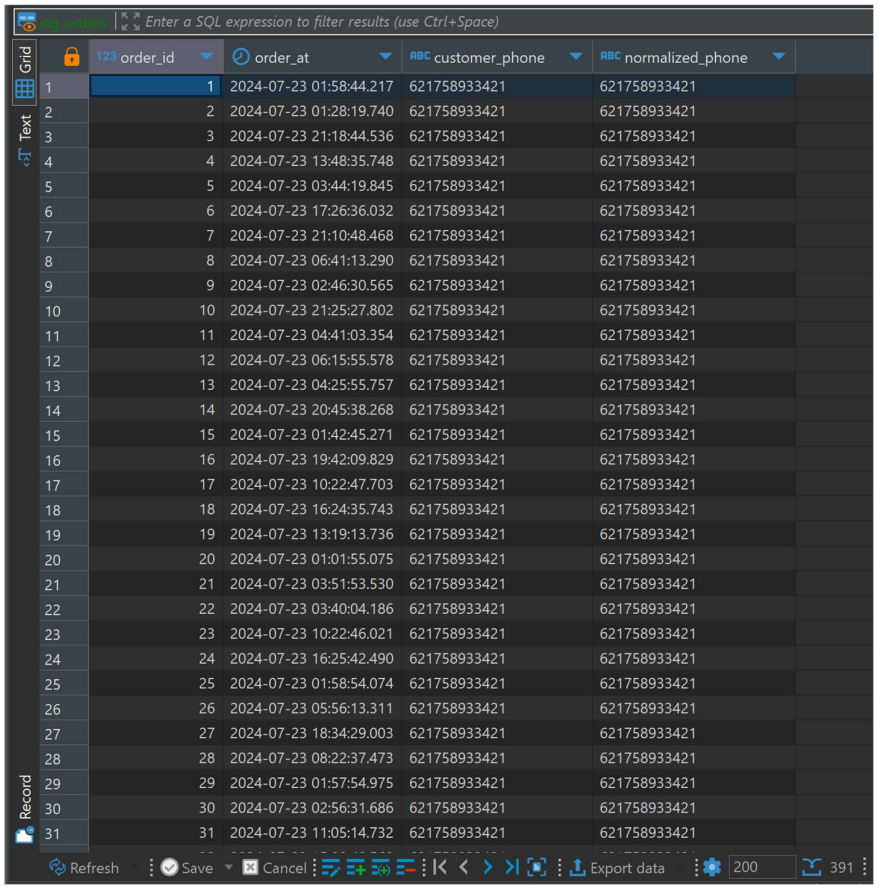
	123 product_id	123 brand_id	ABC product_name	123 product_price
1	1	1	iPhone 13	799.99
2	2	1	MacBook Pro	1,299.99
3	3	2	Galaxy S21	699.99
4	4	3	PlayStation 5	499.99

- **Fungsi:** Mengambil data mentah dari tabel `products` di sumber data dan mengubahnya menjadi tipe data yang diinginkan, kemudian memberi alias kolom yang lebih deskriptif.

2. **stg_orders.sql**

```
select
    order_id::int as order_id
    , order_date::timestamp as order_at
    , customer_phone
    , {{ normalize_phone_number('orders.customer_phone') }} as normalized_phone
from {{ source('store', 'orders') }}
```

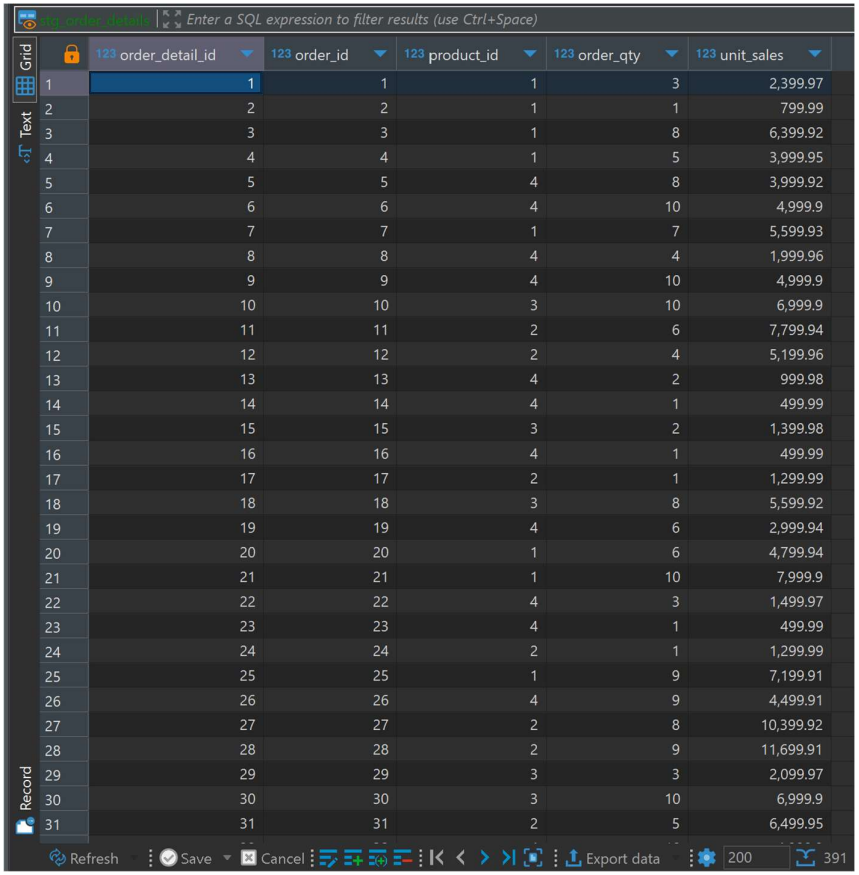
- **Fungsi:** Mengambil data dari tabel `orders`, mengubah tipe data sesuai kebutuhan, dan menormalkan nomor telepon menggunakan macro `normalize_phone_number`.



3. **stg_order_details.sql**

```
select
    order_detail_id::int as order_detail_id
    , order_id::int as order_id
    , product_id::int as product_id
    , quantity as order_qty
    , price::float as unit_sales
from {{ source('store', 'order_details') }}
```

- **Fungsi:** Mengambil data dari tabel `order_details` dan memberi alias serta tipe data yang lebih sesuai untuk tahap berikutnya.

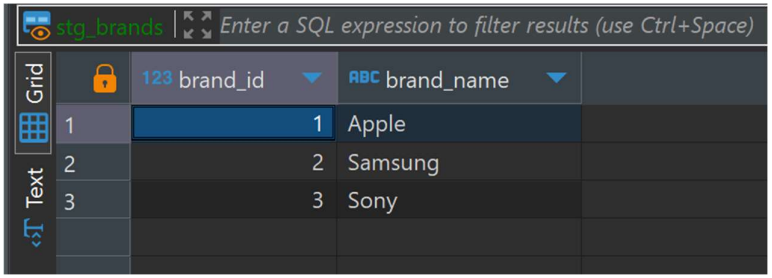


	order_detail_id	order_id	product_id	order_qty	unit_sales
1	1	1	1	3	2,399.97
2	2	2	1	1	799.99
3	3	3	1	8	6,399.92
4	4	4	1	5	3,999.95
5	5	5	4	8	3,999.92
6	6	6	4	10	4,999.9
7	7	7	1	7	5,599.93
8	8	8	4	4	1,999.96
9	9	9	4	10	4,999.9
10	10	10	3	10	6,999.9
11	11	11	2	6	7,799.94
12	12	12	2	4	5,199.96
13	13	13	4	2	999.98
14	14	14	4	1	499.99
15	15	15	3	2	1,399.98
16	16	16	4	1	499.99
17	17	17	2	1	1,299.99
18	18	18	3	8	5,599.92
19	19	19	4	6	2,999.94
20	20	20	1	6	4,799.94
21	21	21	1	10	7,999.9
22	22	22	4	3	1,499.97
23	23	23	4	1	499.99
24	24	24	2	1	1,299.99
25	25	25	1	9	7,199.91
26	26	26	4	9	4,499.91
27	27	27	2	8	10,399.92
28	28	28	2	9	11,699.91
29	29	29	3	3	2,099.97
30	30	30	3	10	6,999.9
31	31	31	2	5	6,499.95

4. **stg_brands.sql**

```
select
    brand_id::int as brand_id
    , name as brand_name
from {{ source('store', 'brands') }}
```

- **Fungsi:** Mengambil data dari tabel `brands` dan mengubah tipe data serta memberi alias yang lebih deskriptif.



5. **stg_schema.yml**

```yaml
version: 2

models:
  - name: stg_brands
    description: "brand staging"
    columns:
      - name: brand_id
        tests:
          - not_null
          - unique

  - name: stg_products
    description: "product staging"
    columns:
      - name: product_id
        tests:
          - not_null
          - unique

      - name: brand_id
        tests:
          - not_null

  - name: stg_orders
    description: "order staging"
    columns:
      - name: order_id
        tests:
          - not_null
```

```
        - unique

- name: stg_order_details
  description: "order detail staging"
  columns:
    - name: order_detail_id
      tests:
        - not_null
        - unique
```

- **Fungsi:** Mendefinisikan skema untuk tabel staging, termasuk deskripsi kolom dan pengujian data seperti `not_null` dan `unique`.

b. Layer Intermediate pada DBT

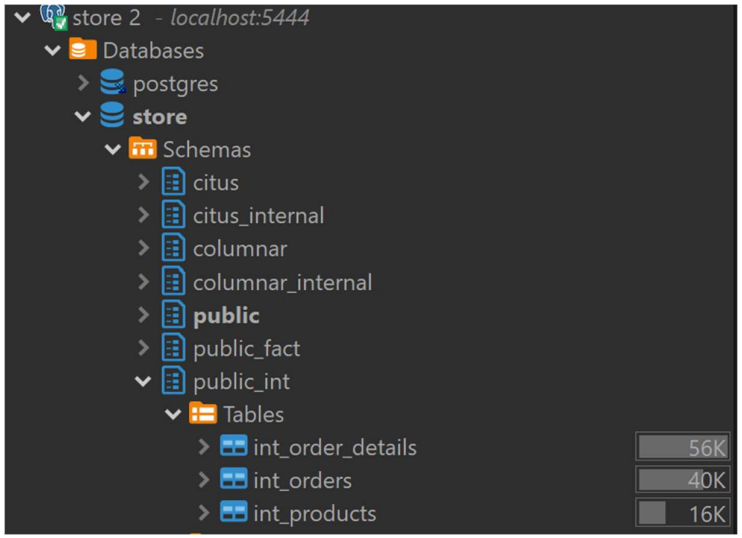
Layer intermediate pada DBT adalah lapisan yang digunakan untuk melakukan transformasi data yang lebih kompleks setelah data diproses melalui layer staging. Model-model di layer intermediate biasanya bertindak sebagai penghubung antara data mentah yang telah dibersihkan dan model-model akhir yang digunakan untuk analisis dan pelaporan. Fungsi utama dari layer intermediate adalah menggabungkan, meringkas, dan memperkaya data untuk keperluan analisis lebih lanjut. Dengan menggunakan layer intermediate, kita dapat meningkatkan modularitas, keterbacaan, dan pengoptimalan kinerja pipeline data.

Fungsi Utama Layer Intermediate

1. **Menggabungkan Data**: Menggabungkan data dari berbagai tabel staging untuk membentuk data yang lebih kaya dan bermakna.
2. **Meringkas Data**: Melakukan agregasi data untuk menghasilkan metrik-metrik yang penting.
3. **Memperkaya Data**: Menambahkan informasi tambahan ke data yang telah dibersihkan, seperti menghitung kolom baru atau menggabungkan data dari sumber lain.
4. **Transformasi Bisnis**: Menerapkan aturan bisnis yang kompleks dan logika transformasi yang tidak dilakukan di layer staging.

`my_project/models/store/_int/`

Intermediate layer ini menggabungkan data dari berbagai tabel staging untuk menyiapkan data yang lebih siap untuk analisis.



1. **int_products.sql**

```sql
select
    products.product_id
    , brands.brand_id
    , brands.brand_name
    , products.product_name
    , products.product_price
from {{ ref('stg_products') }} as products
left join {{ ref('stg_brands') }} as brands
    on products.brand_id = brands.brand_id
```

- **Fungsi:** Menggabungkan data dari tabel `stg_products` dan `stg_brands` untuk mendapatkan informasi produk lengkap dengan nama merek.

int_products | Enter a SQL expression to filter results (use Ctrl+Space)

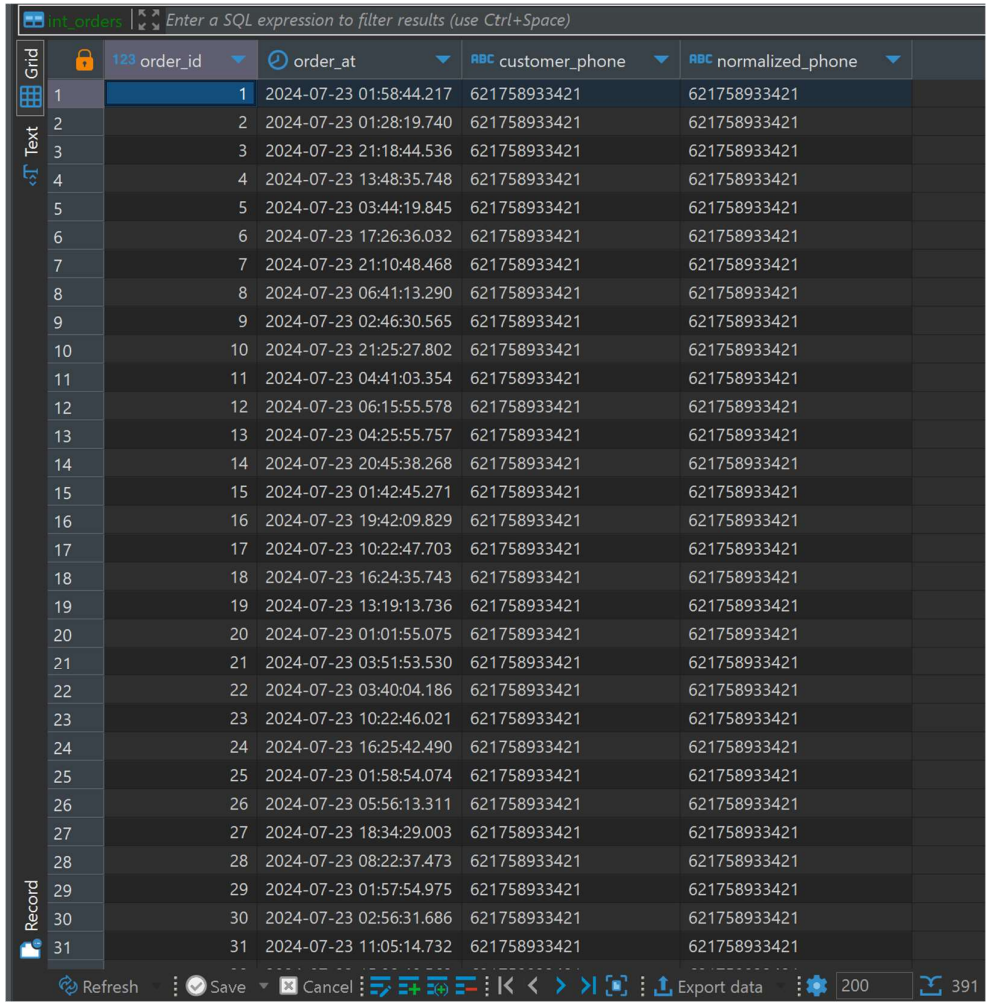
Grid | Text

	product_id	brand_id	brand_name	product_name	product_price
1	1	1	Apple	iPhone 13	799.99
2	2	1	Apple	MacBook Pro	1,299.99
3	3	2	Samsung	Galaxy S21	699.99
4	4	3	Sony	PlayStation 5	499.99

2. **int_orders.sql**

```
select *
from {{ ref('stg_orders') }}
```

- **Fungsi:** Mengambil semua kolom dari tabel `stg_orders` tanpa perubahan.



	order_id	order_at	customer_phone	normalized_phone
1	1	2024-07-23 01:58:44.217	621758933421	621758933421
2	2	2024-07-23 01:28:19.740	621758933421	621758933421
3	3	2024-07-23 21:18:44.536	621758933421	621758933421
4	4	2024-07-23 13:48:35.748	621758933421	621758933421
5	5	2024-07-23 03:44:19.845	621758933421	621758933421
6	6	2024-07-23 17:26:36.032	621758933421	621758933421
7	7	2024-07-23 21:10:48.468	621758933421	621758933421
8	8	2024-07-23 06:41:13.290	621758933421	621758933421
9	9	2024-07-23 02:46:30.565	621758933421	621758933421
10	10	2024-07-23 21:25:27.802	621758933421	621758933421
11	11	2024-07-23 04:41:03.354	621758933421	621758933421
12	12	2024-07-23 06:15:55.578	621758933421	621758933421
13	13	2024-07-23 04:25:55.757	621758933421	621758933421
14	14	2024-07-23 20:45:38.268	621758933421	621758933421
15	15	2024-07-23 01:42:45.271	621758933421	621758933421
16	16	2024-07-23 19:42:09.829	621758933421	621758933421
17	17	2024-07-23 10:22:47.703	621758933421	621758933421
18	18	2024-07-23 16:24:35.743	621758933421	621758933421
19	19	2024-07-23 13:19:13.736	621758933421	621758933421
20	20	2024-07-23 01:01:55.075	621758933421	621758933421
21	21	2024-07-23 03:51:53.530	621758933421	621758933421
22	22	2024-07-23 03:40:04.186	621758933421	621758933421
23	23	2024-07-23 10:22:46.021	621758933421	621758933421
24	24	2024-07-23 16:25:42.490	621758933421	621758933421
25	25	2024-07-23 01:58:54.074	621758933421	621758933421
26	26	2024-07-23 05:56:13.311	621758933421	621758933421
27	27	2024-07-23 18:34:29.003	621758933421	621758933421
28	28	2024-07-23 08:22:37.473	621758933421	621758933421
29	29	2024-07-23 01:57:54.975	621758933421	621758933421
30	30	2024-07-23 02:56:31.686	621758933421	621758933421
31	31	2024-07-23 11:05:14.732	621758933421	621758933421

3. **int_order_details.sql**

```
select
    details.order_detail_id,
    details.order_id,
    orders.order_at,
    {{ normalize_phone_number('orders.customer_phone') }} as normalized_phone,
    case
        when {{ normalize_phone_number('orders.customer_phone') }} like '62%' then 'Indonesia'
        when {{ normalize_phone_number('orders.customer_phone') }} like '91%' then 'India'
        else 'Unknown'
    end as country,
    products.brand_id,
    products.brand_name,
    details.product_id,
    products.product_name,
    details.order_qty,
    details.unit_sales
from {{ ref('stg_order_details') }} as details
left join {{ ref('int_orders') }} as orders
    on details.order_id = orders.order_id
left join {{ ref('int_products') }} as products
    on details.product_id = products.product_id
```

- **Fungsi:** Menggabungkan data dari tabel `stg_order_details`, `int_orders`, dan `int_products` untuk mendapatkan informasi detail pesanan dengan data produk dan informasi negara berdasarkan nomor telepon.

DATA ENGINEER BATCH 4
Mentee: Yovina Silvia
Mentor: Bilal Benefit

alterra academy

int_order_details | Enter a SQL expression to filter results (use Ctrl+Space)

	order_detail_id	order_id	order_at	normalized_phone	country	brand_id	brand_name
1	1	1	2024-07-23 01:58:44.217	621758933421	Indonesia	1	Apple
2	2	2	2024-07-23 01:28:19.740	621758933421	Indonesia	1	Apple
3	3	3	2024-07-23 21:18:44.536	621758933421	Indonesia	1	Apple
4	4	4	2024-07-23 13:48:35.748	621758933421	Indonesia	1	Apple
5	5	5	2024-07-23 03:44:19.845	621758933421	Indonesia	3	Sony
6	6	6	2024-07-23 17:26:36.032	621758933421	Indonesia	3	Sony
7	7	7	2024-07-23 21:10:48.468	621758933421	Indonesia	1	Apple
8	8	8	2024-07-23 06:41:13.290	621758933421	Indonesia	3	Sony
9	9	9	2024-07-23 02:46:30.565	621758933421	Indonesia	3	Sony
10	10	10	2024-07-23 21:25:27.802	621758933421	Indonesia	2	Samsung
11	11	11	2024-07-23 04:41:03.354	621758933421	Indonesia	1	Apple
12	12	12	2024-07-23 06:15:55.578	621758933421	Indonesia	1	Apple
13	13	13	2024-07-23 04:25:55.757	621758933421	Indonesia	3	Sony
14	14	14	2024-07-23 20:45:38.268	621758933421	Indonesia	3	Sony
15	15	15	2024-07-23 01:42:45.271	621758933421	Indonesia	2	Samsung
16	16	16	2024-07-23 19:42:09.829	621758933421	Indonesia	3	Sony
17	17	17	2024-07-23 10:22:47.703	621758933421	Indonesia	1	Apple
18	18	18	2024-07-23 16:24:35.743	621758933421	Indonesia	2	Samsung
19	19	19	2024-07-23 13:19:13.736	621758933421	Indonesia	3	Sony
20	20	20	2024-07-23 01:01:55.075	621758933421	Indonesia	1	Apple
21	21	21	2024-07-23 03:51:53.530	621758933421	Indonesia	1	Apple
22	22	22	2024-07-23 03:40:04.186	621758933421	Indonesia	3	Sony
23	23	23	2024-07-23 10:22:46.021	621758933421	Indonesia	3	Sony
24	24	24	2024-07-23 16:25:42.490	621758933421	Indonesia	1	Apple
25	25	25	2024-07-23 01:58:54.074	621758933421	Indonesia	1	Apple
26	26	26	2024-07-23 05:56:13.311	621758933421	Indonesia	3	Sony
27	27	27	2024-07-23 18:34:29.003	621758933421	Indonesia	1	Apple
28	28	28	2024-07-23 08:22:37.473	621758933421	Indonesia	1	Apple
29	29	29	2024-07-23 01:57:54.975	621758933421	Indonesia	2	Samsung
30	30	30	2024-07-23 02:56:31.686	621758933421	Indonesia	2	Samsung

Refresh | Save | Cancel | Export data | 200 | 391

int_order_details | Enter a SQL expression to filter results (use Ctrl+Space)

	e	country	brand_id	brand_name	product_id	product_name	order_qty	unit_sales
1		Indonesia	1	Apple	1	iPhone 13	3	2,399.97
2		Indonesia	1	Apple	1	iPhone 13	1	799.99
3		Indonesia	1	Apple	1	iPhone 13	8	6,399.92
4		Indonesia	1	Apple	1	iPhone 13	5	3,999.95
5		Indonesia	3	Sony	4	PlayStation 5	8	3,999.92
6		Indonesia	3	Sony	4	PlayStation 5	10	4,999.9
7		Indonesia	1	Apple	1	iPhone 13	7	5,599.93
8		Indonesia	3	Sony	4	PlayStation 5	4	1,999.96
9		Indonesia	3	Sony	4	PlayStation 5	10	4,999.9
10		Indonesia	2	Samsung	3	Galaxy S21	10	6,999.9
11		Indonesia	1	Apple	2	MacBook Pro	6	7,799.94
12		Indonesia	1	Apple	2	MacBook Pro	4	5,199.96
13		Indonesia	3	Sony	4	PlayStation 5	2	999.98
14		Indonesia	3	Sony	4	PlayStation 5	1	499.99
15		Indonesia	2	Samsung	3	Galaxy S21	2	1,399.98
16		Indonesia	3	Sony	4	PlayStation 5	1	499.99
17		Indonesia	1	Apple	2	MacBook Pro	1	1,299.99
18		Indonesia	2	Samsung	3	Galaxy S21	8	5,599.92
19		Indonesia	3	Sony	4	PlayStation 5	6	2,999.94
20		Indonesia	1	Apple	1	iPhone 13	6	4,799.94
21		Indonesia	1	Apple	1	iPhone 13	10	7,999.9
22		Indonesia	3	Sony	4	PlayStation 5	3	1,499.97
23		Indonesia	3	Sony	4	PlayStation 5	1	499.99
24		Indonesia	1	Apple	2	MacBook Pro	1	1,299.99
25		Indonesia	1	Apple	1	iPhone 13	9	7,199.91
26		Indonesia	3	Sony	4	PlayStation 5	9	4,499.91
27		Indonesia	1	Apple	2	MacBook Pro	8	10,399.92
28		Indonesia	1	Apple	2	MacBook Pro	9	11,699.91
29		Indonesia	2	Samsung	3	Galaxy S21	3	2,099.97
30		Indonesia	2	Samsung	3	Galaxy S21	10	6,999.9

Refresh | Save | Cancel | Export data | 200 | 391

4. **int_schema.yml**

```yaml
version: 2

models:
  - name: int_orders
    columns:
      - name: order_id
        tests:
          - not_null
          - unique

  - name: int_products
    columns:
      - name: product_name
        tests:
          - not_null
          - unique

      - name: brand_name
        tests:
          - not_null

  - name: int_order_details
    columns:
      - name: order_detail_id
        tests:
          - not_null
          - unique

    tests:
```

```
      - dbt_expectations.expect_table_row_count_to_equal_other_table:
          compare_model: ref("stg_order_details")
```

- **Fungsi:** Mendefinisikan skema untuk tabel intermediate, termasuk deskripsi kolom dan pengujian data seperti `not_null`, `unique`, dan `expect_table_row_count_to_equal_other_table`.

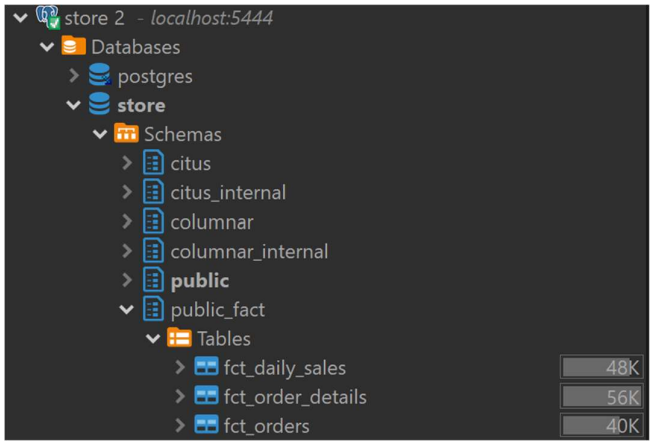
c. Layer Fact pada DBT

Layer fact pada DBT adalah lapisan penting yang bertanggung jawab untuk menyimpan data yang telah diolah dalam bentuk tabel fakta. Tabel fakta berisi data numerik yang dapat diukur dan siap digunakan untuk analisis lebih lanjut. Dengan menggunakan layer fact, kita dapat meningkatkan kinerja query, memastikan konsistensi data, dan mempermudah analisis bisnis. Model fact biasanya berisi data yang telah diringkas dan sering kali dihubungkan dengan tabel dimensi untuk menyediakan konteks tambahan.

Fungsi Utama Layer Fact

1. **Menyimpan Data Agregat**: Tabel fakta biasanya berisi data yang telah diringkas atau diakumulasikan, seperti total penjualan, jumlah pesanan, atau total pendapatan.
2. **Menghubungkan Dimensi**: Tabel fakta sering kali berhubungan dengan tabel dimensi yang menyediakan konteks tambahan seperti produk, waktu, atau pelanggan.
3. **Mendukung Analisis**: Menyediakan data yang siap digunakan untuk analisis lebih lanjut dalam pelaporan atau dashboard bisnis.

`my_project/models/store/_fct/`



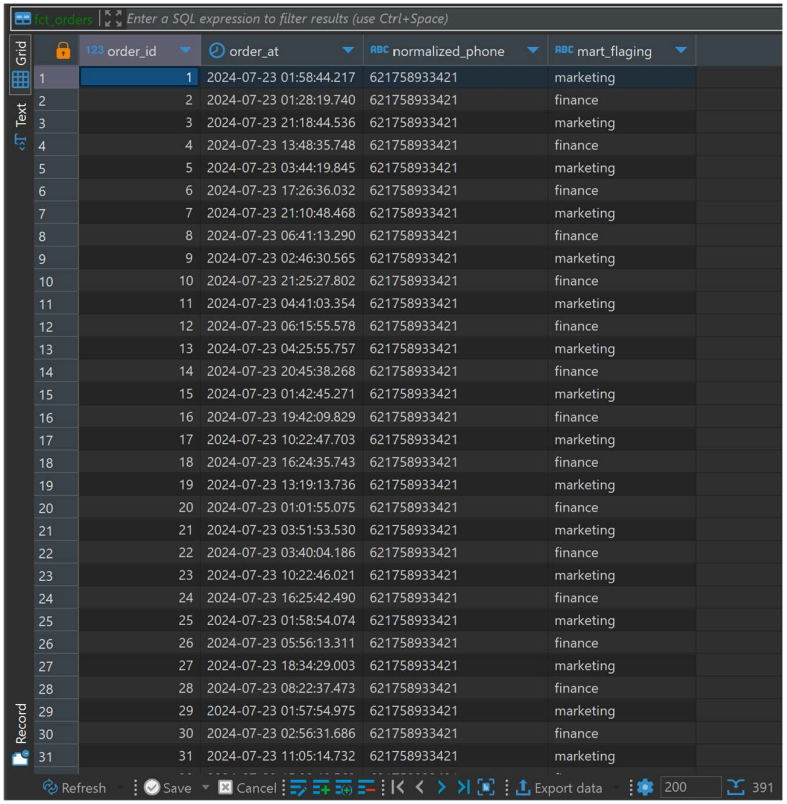
Fact layer ini digunakan untuk membuat tabel fakta yang siap untuk analisis data.

1. **fct_orders.sql**

```
select
    order_id
    , order_at
    , normalized_phone
    , case
        when orders.order_id % 2 != 0 then 'marketing'
        else 'finance'
    end as mart_flaging

from {{ ref('int_orders') }} as orders
```

- **Fungsi:** Mengambil data dari `int_orders` dan menambahkan kolom `mart_flaging` untuk mengkategorikan pesanan sebagai 'marketing' atau 'finance' berdasarkan `order_id`, dan menampilkan kolom normalized_phone yang merupakan nomor telfon customer yang sudah dinormalisasi sehingga mempermudah divisi terit untuk mengelola dan menanfaatkan nomor telfon customer yang sudah dirapikan tersebut.



fct_orders | Enter a SQL expression to filter results (use Ctrl+Space)

	order_id	order_at	normalized_phone	mart_flaging
1	1	2024-07-23 01:58:44.217	621758933421	marketing
2	2	2024-07-23 01:28:19.740	621758933421	finance
3	3	2024-07-23 21:18:44.536	621758933421	marketing
4	4	2024-07-23 13:48:35.748	621758933421	finance
5	5	2024-07-23 03:44:19.845	621758933421	marketing
6	6	2024-07-23 17:26:36.032	621758933421	finance
7	7	2024-07-23 21:10:48.468	621758933421	marketing
8	8	2024-07-23 06:41:13.290	621758933421	finance
9	9	2024-07-23 02:46:30.565	621758933421	marketing
10	10	2024-07-23 21:25:27.802	621758933421	finance
11	11	2024-07-23 04:41:03.354	621758933421	marketing
12	12	2024-07-23 06:15:55.578	621758933421	finance
13	13	2024-07-23 04:25:55.757	621758933421	marketing
14	14	2024-07-23 20:45:38.268	621758933421	finance
15	15	2024-07-23 01:42:45.271	621758933421	marketing
16	16	2024-07-23 19:42:09.829	621758933421	finance
17	17	2024-07-23 10:22:47.703	621758933421	marketing
18	18	2024-07-23 16:24:35.743	621758933421	finance
19	19	2024-07-23 13:19:13.736	621758933421	marketing
20	20	2024-07-23 01:01:55.075	621758933421	finance
21	21	2024-07-23 03:51:53.530	621758933421	marketing
22	22	2024-07-23 03:40:04.186	621758933421	finance
23	23	2024-07-23 10:22:46.021	621758933421	marketing
24	24	2024-07-23 16:25:42.490	621758933421	finance
25	25	2024-07-23 01:58:54.074	621758933421	marketing
26	26	2024-07-23 05:56:13.311	621758933421	finance
27	27	2024-07-23 18:34:29.003	621758933421	marketing
28	28	2024-07-23 08:22:37.473	621758933421	finance
29	29	2024-07-23 01:57:54.975	621758933421	marketing
30	30	2024-07-23 02:56:31.686	621758933421	finance
31	31	2024-07-23 11:05:14.732	621758933421	marketing

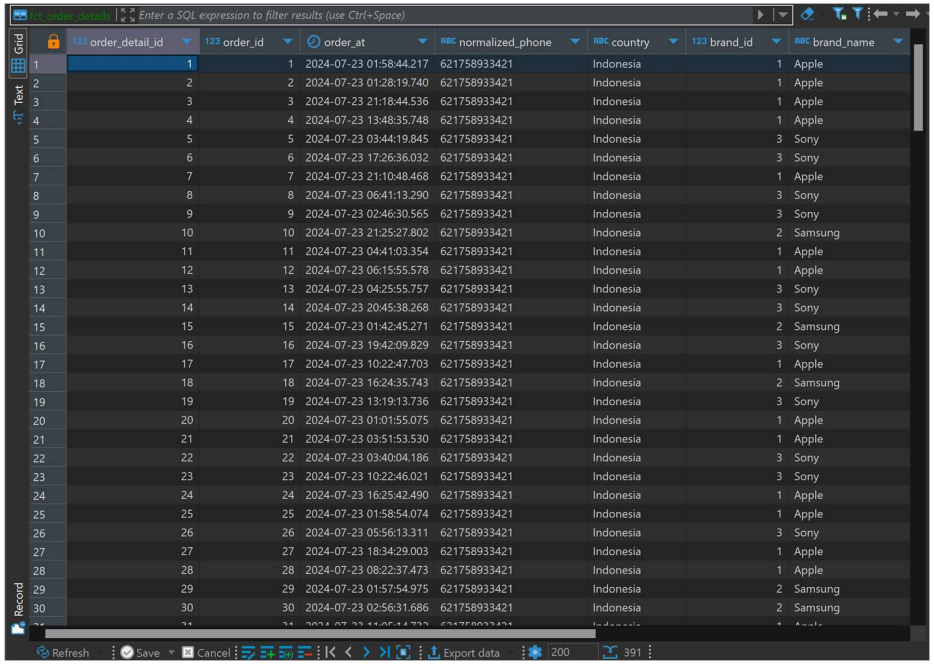
Refresh | Save | Cancel | Export data | 200 | 391

2. **fct_order_details.sql**

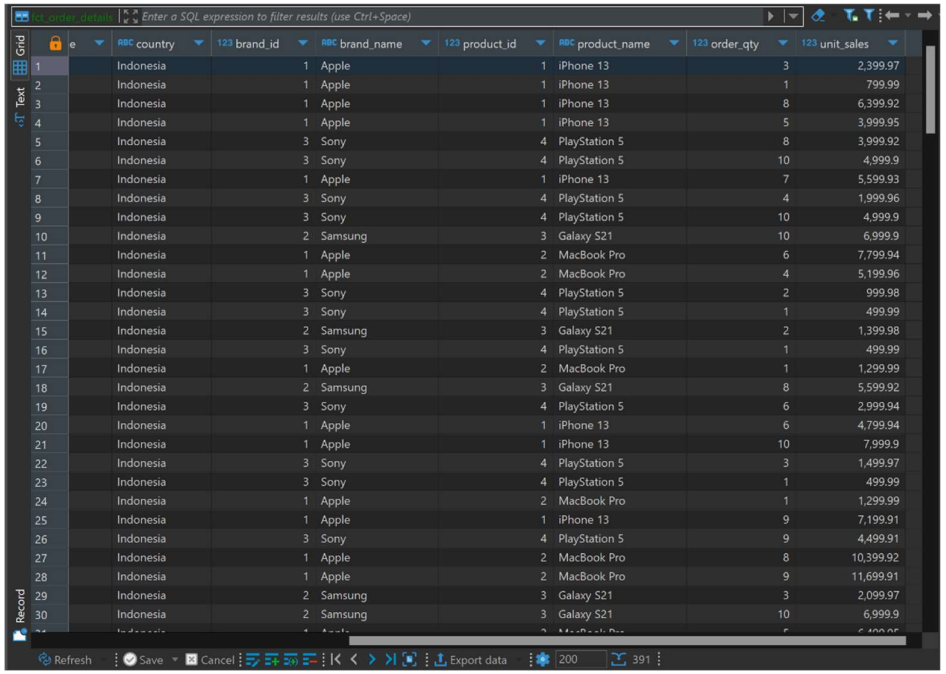
```
select *
from {{ ref('int_order_details') }}
```

- **Fungsi:** Sama seperti `fct_orders.sql`, tetapi mengacu pada detail pesanan.

DATA ENGINEER BATCH 4
Mentee: Yovina Silvia
Mentor: Bilal Benefit



order_detail_id	order_id	order_at	normalized_phone	country	brand_id	brand_name
1	1	2024-07-23 01:58:44.217	621758933421	Indonesia	1	Apple
2	2	2024-07-23 01:28:19.740	621758933421	Indonesia	1	Apple
3	3	2024-07-23 21:18:44.536	621758933421	Indonesia	1	Apple
4	4	2024-07-23 13:48:35.748	621758933421	Indonesia	1	Apple
5	5	2024-07-23 03:44:19.845	621758933421	Indonesia	3	Sony
6	6	2024-07-23 17:26:36.032	621758933421	Indonesia	3	Sony
7	7	2024-07-23 21:10:48.468	621758933421	Indonesia	1	Apple
8	8	2024-07-23 06:41:13.290	621758933421	Indonesia	3	Sony
9	9	2024-07-23 02:46:30.565	621758933421	Indonesia	3	Sony
10	10	2024-07-23 21:25:27.802	621758933421	Indonesia	2	Samsung
11	11	2024-07-23 04:41:03.354	621758933421	Indonesia	1	Apple
12	12	2024-07-23 06:15:55.578	621758933421	Indonesia	1	Apple
13	13	2024-07-23 04:25:55.757	621758933421	Indonesia	3	Sony
14	14	2024-07-23 20:45:38.268	621758933421	Indonesia	3	Sony
15	15	2024-07-23 01:42:45.271	621758933421	Indonesia	2	Samsung
16	16	2024-07-23 19:42:09.829	621758933421	Indonesia	3	Sony
17	17	2024-07-23 10:22:47.703	621758933421	Indonesia	1	Apple
18	18	2024-07-23 16:24:35.743	621758933421	Indonesia	2	Samsung
19	19	2024-07-23 13:19:13.736	621758933421	Indonesia	3	Sony
20	20	2024-07-23 01:01:55.075	621758933421	Indonesia	1	Apple
21	21	2024-07-23 03:51:53.530	621758933421	Indonesia	1	Apple
22	22	2024-07-23 03:40:04.186	621758933421	Indonesia	3	Sony
23	23	2024-07-23 10:22:46.021	621758933421	Indonesia	3	Sony
24	24	2024-07-23 16:25:42.490	621758933421	Indonesia	1	Apple
25	25	2024-07-23 01:58:54.074	621758933421	Indonesia	1	Apple
26	26	2024-07-23 05:56:13.311	621758933421	Indonesia	3	Sony
27	27	2024-07-23 18:34:29.003	621758933421	Indonesia	1	Apple
28	28	2024-07-23 08:22:37.473	621758933421	Indonesia	1	Apple
29	29	2024-07-23 01:57:54.975	621758933421	Indonesia	2	Samsung
30	30	2024-07-23 02:56:31.686	621758933421	Indonesia	2	Samsung

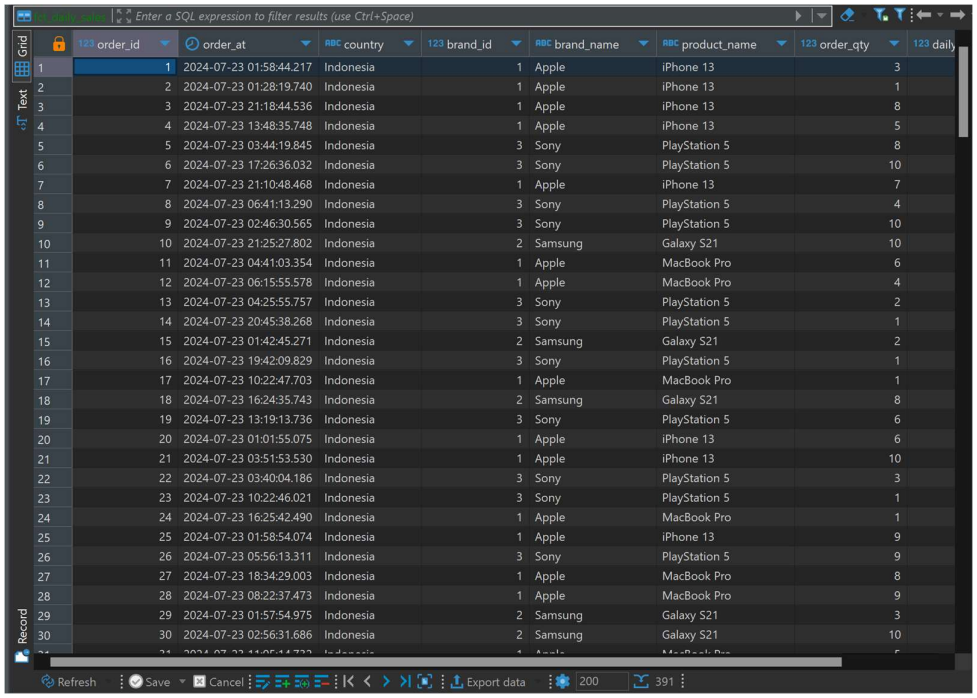


country	brand_id	brand_name	product_id	product_name	order_qty	unit_sales
Indonesia	1	Apple	1	iPhone 13	3	2,399.97
Indonesia	1	Apple	1	iPhone 13	1	799.99
Indonesia	1	Apple	1	iPhone 13	8	6,399.92
Indonesia	1	Apple	1	iPhone 13	5	3,999.95
Indonesia	3	Sony	4	PlayStation 5	8	3,999.92
Indonesia	3	Sony	4	PlayStation 5	10	4,999.9
Indonesia	1	Apple	1	iPhone 13	7	5,599.93
Indonesia	3	Sony	4	PlayStation 5	4	1,999.96
Indonesia	3	Sony	4	PlayStation 5	10	4,999.9
Indonesia	2	Samsung	3	Galaxy S21	10	6,999.9
Indonesia	1	Apple	2	MacBook Pro	6	7,799.94
Indonesia	1	Apple	2	MacBook Pro	4	5,199.96
Indonesia	3	Sony	4	PlayStation 5	2	999.98
Indonesia	3	Sony	4	PlayStation 5	1	499.99
Indonesia	2	Samsung	3	Galaxy S21	2	1,399.98
Indonesia	3	Sony	4	PlayStation 5	1	499.99
Indonesia	1	Apple	2	MacBook Pro	1	1,299.99
Indonesia	2	Samsung	3	Galaxy S21	8	5,599.92
Indonesia	3	Sony	4	PlayStation 5	6	2,999.94
Indonesia	1	Apple	1	iPhone 13	6	4,799.94
Indonesia	1	Apple	1	iPhone 13	10	7,999.9
Indonesia	3	Sony	4	PlayStation 5	3	1,499.97
Indonesia	3	Sony	4	PlayStation 5	1	499.99
Indonesia	1	Apple	2	MacBook Pro	1	1,299.99
Indonesia	1	Apple	1	iPhone 13	9	7,199.91
Indonesia	3	Sony	4	PlayStation 5	9	4,499.91
Indonesia	1	Apple	2	MacBook Pro	8	10,399.92
Indonesia	1	Apple	2	MacBook Pro	9	11,699.91
Indonesia	2	Samsung	3	Galaxy S21	3	2,099.97
Indonesia	2	Samsung	3	Galaxy S21	10	6,999.9

3. **fct_sales_daily.sql**

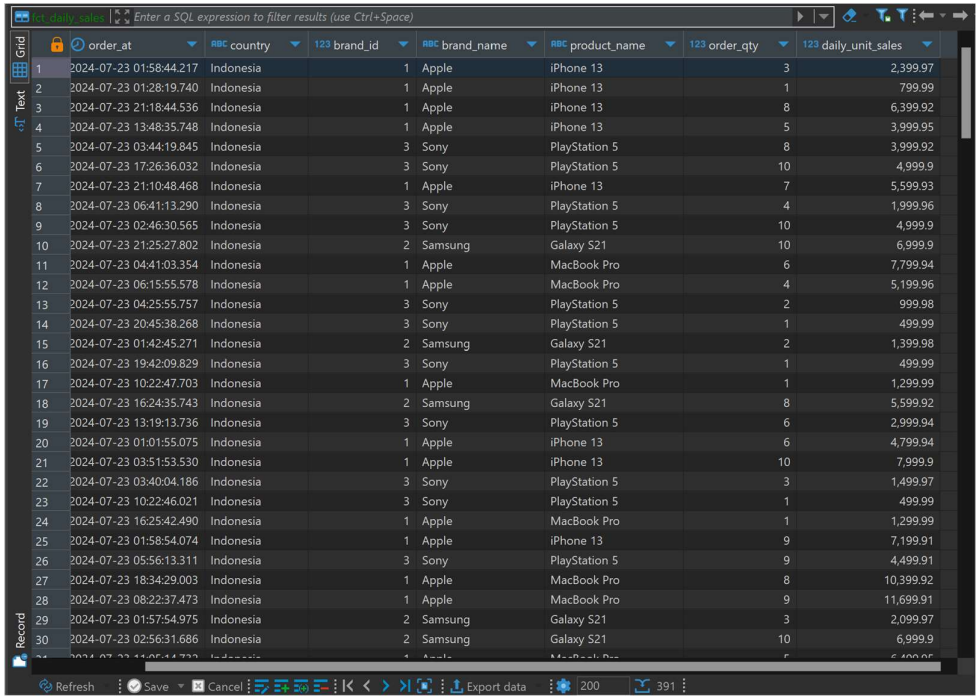
```
select
    order_id,
    order_at,
    country,
    brand_id,
    brand_name,
    product_name,
    order_qty,
    unit_sales as daily_unit_sales
from {{ ref('int_order_details') }}
```

- **Fungsi:** Mengambil data dari `int_order_details` untuk menghasilkan laporan penjualan harian.



	order_id	order_at	country	brand_id	brand_name	product_name	order_qty	daily
1	1	2024-07-23 01:58:44.217	Indonesia	1	Apple	iPhone 13	3	
2	2	2024-07-23 01:28:19.740	Indonesia	1	Apple	iPhone 13	1	
3	3	2024-07-23 21:18:44.536	Indonesia	1	Apple	iPhone 13	8	
4	4	2024-07-23 13:48:35.748	Indonesia	1	Apple	iPhone 13	5	
5	5	2024-07-23 03:44:19.845	Indonesia	3	Sony	PlayStation 5	8	
6	6	2024-07-23 17:26:36.032	Indonesia	3	Sony	PlayStation 5	10	
7	7	2024-07-23 21:10:48.468	Indonesia	1	Apple	iPhone 13	7	
8	8	2024-07-23 06:41:13.290	Indonesia	3	Sony	PlayStation 5	4	
9	9	2024-07-23 02:46:30.565	Indonesia	3	Sony	PlayStation 5	10	
10	10	2024-07-23 21:25:27.802	Indonesia	2	Samsung	Galaxy S21	10	
11	11	2024-07-23 04:41:03.354	Indonesia	1	Apple	MacBook Pro	6	
12	12	2024-07-23 06:15:55.578	Indonesia	1	Apple	MacBook Pro	4	
13	13	2024-07-23 04:25:55.757	Indonesia	3	Sony	PlayStation 5	2	
14	14	2024-07-23 20:45:38.268	Indonesia	3	Sony	PlayStation 5	1	
15	15	2024-07-23 01:42:45.271	Indonesia	2	Samsung	Galaxy S21	2	
16	16	2024-07-23 19:42:09.829	Indonesia	3	Sony	PlayStation 5	1	
17	17	2024-07-23 10:22:47.703	Indonesia	1	Apple	MacBook Pro	1	
18	18	2024-07-23 16:24:35.743	Indonesia	2	Samsung	Galaxy S21	8	
19	19	2024-07-23 13:19:13.736	Indonesia	3	Sony	PlayStation 5	6	
20	20	2024-07-23 01:01:55.075	Indonesia	1	Apple	iPhone 13	6	
21	21	2024-07-23 03:51:53.530	Indonesia	1	Apple	iPhone 13	10	
22	22	2024-07-23 03:40:04.186	Indonesia	3	Sony	PlayStation 5	3	
23	23	2024-07-23 10:22:46.021	Indonesia	3	Sony	PlayStation 5	1	
24	24	2024-07-23 16:25:42.490	Indonesia	1	Apple	MacBook Pro	1	
25	25	2024-07-23 01:58:54.074	Indonesia	1	Apple	iPhone 13	9	
26	26	2024-07-23 05:56:13.311	Indonesia	3	Sony	PlayStation 5	9	
27	27	2024-07-23 18:34:29.003	Indonesia	1	Apple	MacBook Pro	8	
28	28	2024-07-23 08:22:37.473	Indonesia	1	Apple	MacBook Pro	9	
29	29	2024-07-23 01:57:54.975	Indonesia	2	Samsung	Galaxy S21	3	
30	30	2024-07-23 02:56:31.686	Indonesia	2	Samsung	Galaxy S21	10	

DATA ENGINEER BATCH 4
Mentee: Yovina Silvia
Mentor: Bilal Benefit



	order_at	country	brand_id	brand_name	product_name	order_qty	daily_unit_sales
1	2024-07-23 01:58:44.217	Indonesia	1	Apple	iPhone 13	3	2,399.97
2	2024-07-23 01:28:19.740	Indonesia	1	Apple	iPhone 13	1	799.99
3	2024-07-23 21:18:44.536	Indonesia	1	Apple	iPhone 13	8	6,399.92
4	2024-07-23 13:48:35.748	Indonesia	1	Apple	iPhone 13	5	3,999.95
5	2024-07-23 03:44:19.845	Indonesia	3	Sony	PlayStation 5	8	3,999.92
6	2024-07-23 17:26:36.032	Indonesia	3	Sony	PlayStation 5	10	4,999.9
7	2024-07-23 21:10:48.468	Indonesia	1	Apple	iPhone 13	7	5,599.93
8	2024-07-23 06:41:13.290	Indonesia	3	Sony	PlayStation 5	4	1,999.96
9	2024-07-23 02:46:30.565	Indonesia	3	Sony	PlayStation 5	10	4,999.9
10	2024-07-23 21:25:27.802	Indonesia	2	Samsung	Galaxy S21	10	6,999.9
11	2024-07-23 04:41:03.354	Indonesia	1	Apple	MacBook Pro	6	7,799.94
12	2024-07-23 06:15:55.578	Indonesia	1	Apple	MacBook Pro	4	5,199.96
13	2024-07-23 04:25:55.757	Indonesia	3	Sony	PlayStation 5	2	999.98
14	2024-07-23 20:45:38.268	Indonesia	3	Sony	PlayStation 5	1	499.99
15	2024-07-23 01:42:45.271	Indonesia	2	Samsung	Galaxy S21	2	1,399.98
16	2024-07-23 19:42:09.829	Indonesia	3	Sony	PlayStation 5	1	499.99
17	2024-07-23 10:22:47.703	Indonesia	1	Apple	MacBook Pro	1	1,299.99
18	2024-07-23 16:24:35.743	Indonesia	2	Samsung	Galaxy S21	8	5,599.92
19	2024-07-23 13:19:13.736	Indonesia	3	Sony	PlayStation 5	6	2,999.94
20	2024-07-23 01:01:55.075	Indonesia	1	Apple	iPhone 13	6	4,799.94
21	2024-07-23 03:51:53.530	Indonesia	1	Apple	iPhone 13	10	7,999.9
22	2024-07-23 03:40:04.186	Indonesia	3	Sony	PlayStation 5	3	1,499.97
23	2024-07-23 10:22:46.021	Indonesia	3	Sony	PlayStation 5	1	499.99
24	2024-07-23 16:25:42.490	Indonesia	1	Apple	MacBook Pro	1	1,299.99
25	2024-07-23 01:58:54.074	Indonesia	1	Apple	iPhone 13	9	7,199.91
26	2024-07-23 05:56:13.311	Indonesia	3	Sony	PlayStation 5	9	4,499.91
27	2024-07-23 18:34:29.003	Indonesia	1	Apple	MacBook Pro	8	10,399.92
28	2024-07-23 08:22:37.473	Indonesia	1	Apple	MacBook Pro	9	11,699.91
29	2024-07-23 01:57:54.975	Indonesia	2	Samsung	Galaxy S21	3	2,099.97
30	2024-07-23 02:56:31.686	Indonesia	2	Samsung	Galaxy S21	10	6,999.9

4. **fct_schema.yml**

```yaml
version: 2

models:
  - name: fct_orders
    columns:
      - name: order_id
        tests:
          - not_null
          - unique
    tests:
      - dbt_expectations.expect_table_row_count_to_equal_other_table:
          compare_model: ref("int_order_details")

  - name: fct_order_details
    columns:
      - name: order_detail_id
        tests:
          - not_null
          - unique
    tests:
      - dbt_utils.unique_combination_of_columns:
          combination_of_columns:
            - order_id
            - product_id
```

- **Fungsi:** Mendefinisikan skema untuk tabel fakta, termasuk deskripsi kolom dan pengujian data seperti `not_null`, `unique`, dan `unique_combination_of_columns`.

d. Layer Mart pada DBT

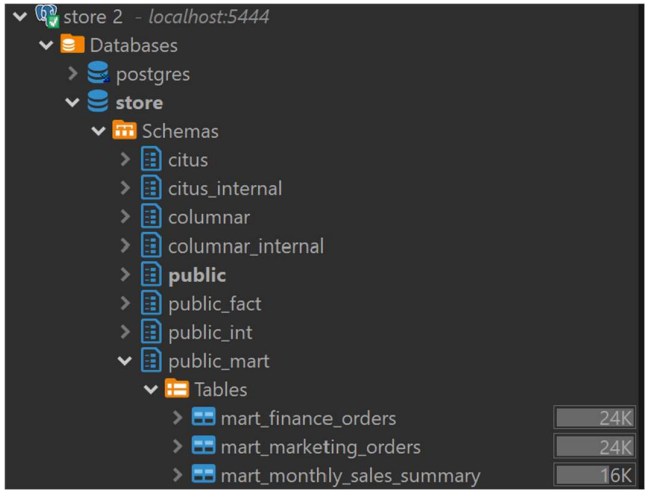
Layer mart pada DBT adalah lapisan terakhir dalam pipeline transformasi data yang bertujuan untuk menyusun data dalam bentuk yang siap digunakan untuk analisis bisnis dan pelaporan. Layer ini berfungsi untuk mengorganisir data yang telah diolah dalam bentuk yang mudah diakses dan dipahami oleh pengguna akhir, seperti analis bisnis, manajer, atau alat BI (Business Intelligence).

Fungsi Utama Layer Mart

1. **Menyediakan Data Siap Analisis**: Mengatur dan menyajikan data yang siap untuk digunakan dalam analisis bisnis dan pelaporan.
2. **Menyederhanakan Akses Data**: Menyederhanakan akses ke data dengan menyediakan struktur data yang dirancang untuk kebutuhan analisis tertentu.
3. **Menggabungkan Data dari Berbagai Sumber**: Menggabungkan data dari berbagai sumber dan model intermediate untuk memberikan pandangan yang komprehensif.
4. **Memfasilitasi Pelaporan dan Dashboarding**: Menyediakan data dalam format yang mudah diintegrasikan dengan alat BI dan dashboard.

`my_project/models/store/_mart/`

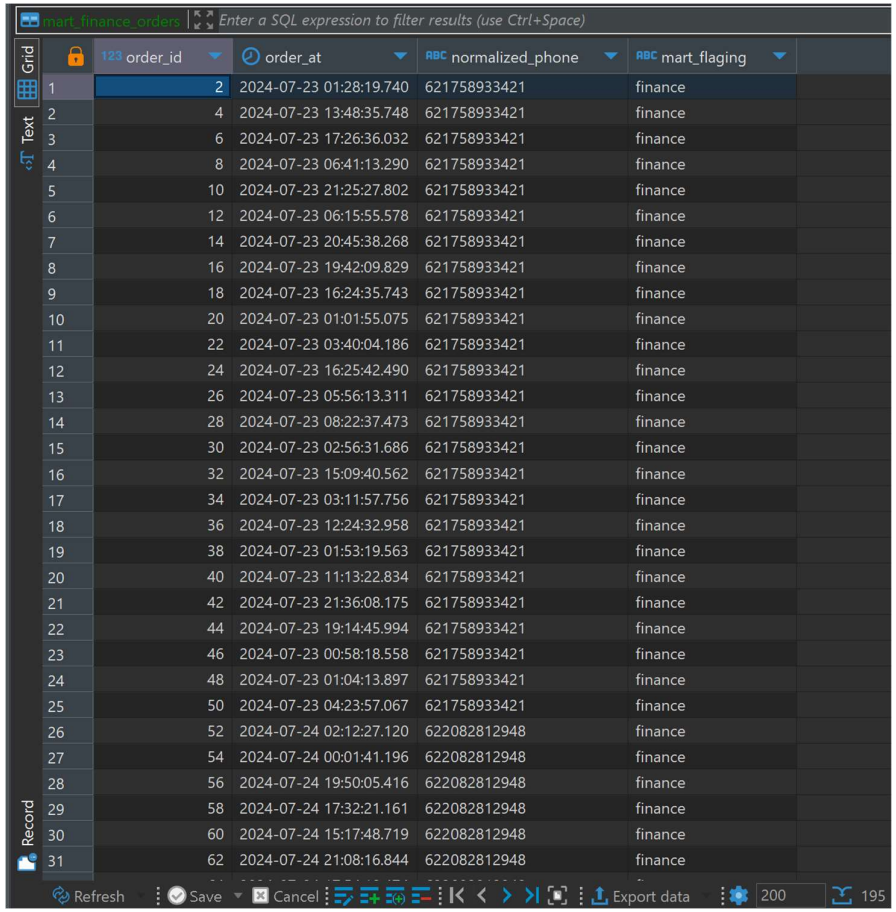
Mart layer ini digunakan untuk membuat tabel mart yang siap untuk kebutuhan analisis lebih spesifik.



1. **mart_finance_orders.sql**

```
select *
from {{ ref('fct_orders') }}
where mart_flaging = 'finance'
```

- **Fungsi:** Mengambil semua data dari `fct_orders` yang dikategorikan sebagai 'finance'.

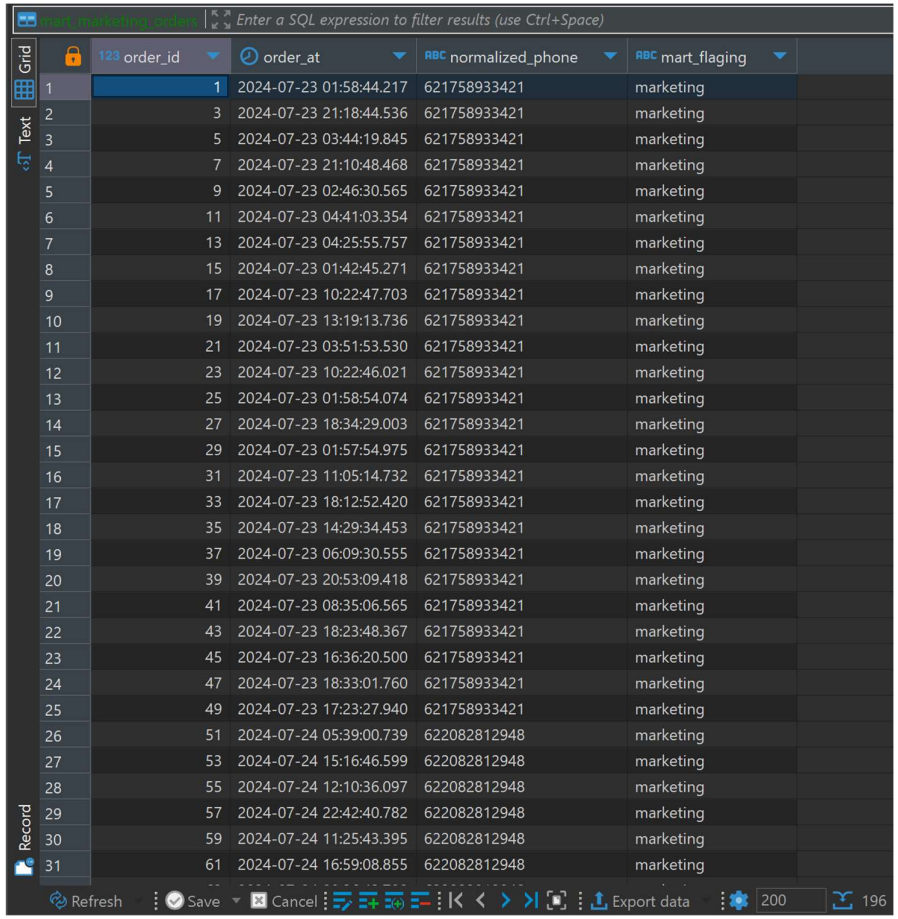


	order_id	order_at	normalized_phone	mart_flaging
1	2	2024-07-23 01:28:19.740	621758933421	finance
2	4	2024-07-23 13:48:35.748	621758933421	finance
3	6	2024-07-23 17:26:36.032	621758933421	finance
4	8	2024-07-23 06:41:13.290	621758933421	finance
5	10	2024-07-23 21:25:27.802	621758933421	finance
6	12	2024-07-23 06:15:55.578	621758933421	finance
7	14	2024-07-23 20:45:38.268	621758933421	finance
8	16	2024-07-23 19:42:09.829	621758933421	finance
9	18	2024-07-23 16:24:35.743	621758933421	finance
10	20	2024-07-23 01:01:55.075	621758933421	finance
11	22	2024-07-23 03:40:04.186	621758933421	finance
12	24	2024-07-23 16:25:42.490	621758933421	finance
13	26	2024-07-23 05:56:13.311	621758933421	finance
14	28	2024-07-23 08:22:37.473	621758933421	finance
15	30	2024-07-23 02:56:31.686	621758933421	finance
16	32	2024-07-23 15:09:40.562	621758933421	finance
17	34	2024-07-23 03:11:57.756	621758933421	finance
18	36	2024-07-23 12:24:32.958	621758933421	finance
19	38	2024-07-23 01:53:19.563	621758933421	finance
20	40	2024-07-23 11:13:22.834	621758933421	finance
21	42	2024-07-23 21:36:08.175	621758933421	finance
22	44	2024-07-23 19:14:45.994	621758933421	finance
23	46	2024-07-23 00:58:18.558	621758933421	finance
24	48	2024-07-23 01:04:13.897	621758933421	finance
25	50	2024-07-23 04:23:57.067	621758933421	finance
26	52	2024-07-24 02:12:27.120	622082812948	finance
27	54	2024-07-24 00:01:41.196	622082812948	finance
28	56	2024-07-24 19:50:05.416	622082812948	finance
29	58	2024-07-24 17:32:21.161	622082812948	finance
30	60	2024-07-24 15:17:48.719	622082812948	finance
31	62	2024-07-24 21:08:16.844	622082812948	finance

2. **mart_marketing_orders.sql**

```
select *
from {{ ref('fct_orders') }}
where mart_flaging = 'marketing'
```

- **Fungsi:** Mengambil semua data dari `fct_orders` yang dikategorikan sebagai 'marketing'.



	order_id	order_at	normalized_phone	mart_flaging
1	1	2024-07-23 01:58:44.217	621758933421	marketing
2	3	2024-07-23 21:18:44.536	621758933421	marketing
3	5	2024-07-23 03:44:19.845	621758933421	marketing
4	7	2024-07-23 21:10:48.468	621758933421	marketing
5	9	2024-07-23 02:46:30.565	621758933421	marketing
6	11	2024-07-23 04:41:03.354	621758933421	marketing
7	13	2024-07-23 04:25:55.757	621758933421	marketing
8	15	2024-07-23 01:42:45.271	621758933421	marketing
9	17	2024-07-23 10:22:47.703	621758933421	marketing
10	19	2024-07-23 13:19:13.736	621758933421	marketing
11	21	2024-07-23 03:51:53.530	621758933421	marketing
12	23	2024-07-23 10:22:46.021	621758933421	marketing
13	25	2024-07-23 01:58:54.074	621758933421	marketing
14	27	2024-07-23 18:34:29.003	621758933421	marketing
15	29	2024-07-23 01:57:54.975	621758933421	marketing
16	31	2024-07-23 11:05:14.732	621758933421	marketing
17	33	2024-07-23 18:12:52.420	621758933421	marketing
18	35	2024-07-23 14:29:34.453	621758933421	marketing
19	37	2024-07-23 06:09:30.555	621758933421	marketing
20	39	2024-07-23 20:53:09.418	621758933421	marketing
21	41	2024-07-23 08:35:06.565	621758933421	marketing
22	43	2024-07-23 18:23:48.367	621758933421	marketing
23	45	2024-07-23 16:36:20.500	621758933421	marketing
24	47	2024-07-23 18:33:01.760	621758933421	marketing
25	49	2024-07-23 17:23:27.940	621758933421	marketing
26	51	2024-07-24 05:39:00.739	622082812948	marketing
27	53	2024-07-24 15:16:46.599	622082812948	marketing
28	55	2024-07-24 12:10:36.097	622082812948	marketing
29	57	2024-07-24 22:42:40.782	622082812948	marketing
30	59	2024-07-24 11:25:43.395	622082812948	marketing
31	61	2024-07-24 16:59:08.855	622082812948	marketing

3. **mart_monthly_sales_summary.sql**

```sql
select
    date_trunc('month', order_at) as month,
    brand_id,
    brand_name,
    product_name,
    SUM(order_qty) as total_order_qty,
    sum(unit_sales) as total_unit_sales,
    sum(unit_sales * order_qty) as total_amount
from {{ ref('fct_order_details') }}
group by 1, 2, 3, 4
order by 1, 2, 3
```

- **Fungsi:** Mengambil data dari `fct_sales_daily` dan mengelompokkan berdasarkan bulan, brand_id, brand_name, dan product_name untuk mendapatkan ringkasan penjualan bulanan.

mart_monthly_sales_summary | Enter a SQL expression to filter results (use Ctrl+Space)

Grid | Text

	month	brand_id	brand_name	product_name	total_order_qty	total_unit_sales	total_amount
1	2024-07-01 00:00:00.000	1	Apple	iPhone 13	399	319,196.0099999999	2,243,971.9499999997
2	2024-07-01 00:00:00.000	1	Apple	MacBook Pro	325	422,496.7499999998	2,798,878.4699999993
3	2024-07-01 00:00:00.000	2	Samsung	Galaxy S21	281	196,697.19	1,344,680.7899999996
4	2024-07-01 00:00:00.000	3	Sony	PlayStation 5	435	217,495.65	1,542,469.1500000001
5	2024-08-01 00:00:00.000	1	Apple	iPhone 13	105	83,998.95	506,393.67
6	2024-08-01 00:00:00.000	1	Apple	MacBook Pro	223	289,897.77	2,125,483.65
7	2024-08-01 00:00:00.000	2	Samsung	Galaxy S21	194	135,798.06	1,006,585.6199999999
8	2024-08-01 00:00:00.000	3	Sony	PlayStation 5	145	72,498.55	442,491.1499999999

e. Macros dalam DBT

Macros dalam DBT adalah alat yang sangat kuat yang memungkinkan untuk menulis kode yang lebih modular, dapat digunakan kembali, dan mudah dipelihara. Dengan menggunakan macros, kita dapat menyederhanakan logika yang kompleks, mengurangi duplikasi kode, dan mempercepat pengembangan proyek DBT. Macros memungkinkan untuk memisahkan logika transformasi data ke dalam fungsi-fungsi kecil yang dapat digunakan kembali di berbagai model, meningkatkan keterbacaan dan keteraturan proyek yang sedang dibuat.

Fungsi Utama Macros

1. **Menyederhanakan Kode**: Mengurangi duplikasi kode dengan membuat potongan kode yang dapat digunakan kembali di berbagai model.
2. **Mempercepat Pengembangan**: Mengotomatiskan tugas-tugas yang berulang dan kompleks sehingga mempercepat proses pengembangan.
3. **Meningkatkan Keterbacaan**: Memisahkan logika yang kompleks ke dalam fungsi-fungsi kecil yang lebih mudah dipahami.
4. **Membuat Kode Lebih Modular**: Memungkinkan pengelolaan yang lebih baik dan pemeliharaan kode yang lebih mudah.

`my_project/macros/`

Folder ini digunakan untuk menyimpan macro custom yang bisa digunakan di dalam model dbt.

1. **phone_normalization.sql**

```
{% macro normalize_phone_number(column_name) %}
    ltrim({{ column_name }}, '+')
{% endmacro %}
```

- **Fungsi:** Macro untuk menormalkan nomor telepon dengan menghilangkan tanda hubung dan spasi.

f. `dbt_project.yml`

File `dbt_project.yml` adalah pusat pengaturan proyek DBT. File ini mengatur struktur proyek, jalur model dan macros, serta perilaku build untuk setiap model. Dengan memahami dan mengonfigurasi dbt_project.yml dengan benar, kita dapat mengelola proyek DBT dengan lebih efektif dan efisien, memastikan bahwa semua aspek proyek diatur dengan baik dan dapat dikelola dengan mudah.

`my_project/dbt_project.yml`

```yaml
name: 'my_project'
version: '1.0.0'
config-version: 2

profile: 'my_project'

model-paths: ["models"]
analysis-paths: ["analyses"]
test-paths: ["tests"]
seed-paths: ["seeds"]
macro-paths: ["macros"]
snapshot-paths: ["snapshots"]

clean-targets:
  - "target"
  - "dbt_packages"


models:
  my_project:
    store:
      _stg:
        +materialized: view
```

```
      +schema: stg
      +database: store
    _int:
      +materialized: table
      +schema: int
      +database: store
    _fct:
      +materialized: table
      +schema: fact
      +database: store
    _mart:
      +materialized: table
      +schema: mart
      +database: store
```

- **Fungsi:** Mengatur berbagai path untuk model, analisis, test, seeds, macro, dan snapshot. Juga mengatur bagaimana model harus dimaterialisasikan dan di mana schema serta database-nya berada.

g. packages.yml

File packages.yml adalah komponen penting dalam proyek DBT untuk mengelola dependensi eksternal. Dengan mendefinisikan paket-paket yang diperlukan dalam file ini, kita dapat memastikan bahwa proyek tetap modular, dapat dipelihara dengan mudah, dan memanfaatkan fungsionalitas yang sudah ada dari komunitas DBT. Setelah mendefinisikan paket-paket ini, kita hanya perlu menjalankan dbt deps untuk menginstalnya, membuat manajemen dependensi menjadi sederhana dan efisien.

my_project/packages.yml

File ini digunakan untuk mendefinisikan paket tambahan yang ingin digunakan dalam proyek dbt kita.

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

```
    version: 1.2.0
  - package: calogica/dbt_expectations
    version: 0.10.3
```

- **Fungsi:** Menyediakan utilitas tambahan dari dbt dan paket ekspektasi untuk pengujian data.

my_project/package-lock.yml

File ini berisi dependensi yang telah dikunci oleh dbt dan biasanya tidak perlu diubah.

```
# This file is auto-generated by dbt. Do not edit.
```

h. sources.yml

File sources.yml dalam proyek DBT digunakan untuk mendefinisikan sumber data (sources) yang berada di luar DBT dan yang akan digunakan dalam model DBT. Sumber data ini bisa berupa tabel atau view yang ada di database yang ada dan yang belum dihasilkan oleh model DBT. Dengan mendefinisikan sumber data dalam file sources.yml, kita dapat dengan mudah mereferensikannya dalam model DBT dan juga memanfaatkan fitur dokumentasi dan pengujian bawaan DBT.

my_project/sources.yml

File ini mendefinisikan sumber data yang akan digunakan dalam proyek dbt.

```
version: 2

sources:
  - name: store
    database: store
    schema: public  # Skema tempat tabel sumber berada
    tables:
      - name: brands
      - name: products
```

```
      - name: orders
      - name: order_details
```

- **Fungsi:** Mendefinisikan sumber data `store` yang terdiri dari tabel `products`, `orders`, `order_details`, dan `brands`.

i. profiles.yml

File profiles.yml adalah file konfigurasi di DBT yang digunakan untuk mengatur koneksi ke database. File ini memungkinkan kita untuk mengelola berbagai lingkungan dan target koneksi, seperti lingkungan pengembangan (development), pengujian (test), dan produksi (production). Dengan profiles.yml, kita dapat menentukan detail koneksi untuk setiap lingkungan, termasuk kredensial, host, port, dan database yang akan digunakan.

`dbt-profiles/profiles.yml`

File ini mengatur profil koneksi untuk proyek dbt.

```
my_project:
  outputs:

    dev:
      type: postgres
      threads: 1
      host: localhost
      port: 5444
      user: postgres
      pass: pass
      dbname: store
      schema: public

  target: dev
```

- **Fungsi:** Mendefinisikan profil `my_project` dengan output `dev` yang terhubung ke database PostgreSQL.

j. `docker.compose.yml`

File ini mengatur Docker untuk menjalankan database PostgreSQL.

```yaml
version: '3'

services:

  master:
    container_name: "${COMPOSE_PROJECT_NAME:-citus}_master"
    image: "citusdata/citus:12.0.0"
    ports:
      - "${COORDINATOR_EXTERNAL_PORT:-5444}:5432"
    labels:
      - "com.citusdata.role=Master"
    environment: &AUTH
      - POSTGRES_PASSWORD=pass
      - POSTGRES_USER=postgres
      - POSTGRES_DB=store
      - PGUSER=postgres
      - POSTGRES_HOST_AUTH_METHOD=trust
    networks:
      - postgres-network
    volumes:
      - ./citus-db-data/:/var/lib/postgresql/data/
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql

  worker:
    image: "citusdata/citus:12.0.0"
```

```
    labels:
      - "com.citusdata.role=Worker"
    depends_on:
      - manager
    environment: *AUTH
    command: "/wait-for-manager.sh"
    volumes:
      - ./citus-healthcheck:/healthcheck

  manager:
    container_name: "${COMPOSE_PROJECT_NAME:-citus}_manager"
    image: "citusdata/membership-manager:0.3.0"
    volumes:
      - "${DOCKER_SOCK:-/var/run/docker.sock}:/var/run/docker.sock"
      - ./citus-healthcheck:/healthcheck
    depends_on:
      - master
    environment: *AUTH

networks:
  postgres-network:
    driver: bridge
```

- **Fungsi:** Menyediakan layanan PostgreSQL menggunakan Docker.

k. `requirements.txt`

File ini mendefinisikan dependensi Python untuk proyek dbt.

```
dbt-core==1.7.0
dbt-extractor==0.5.0
```

```
dbt-postgres==1.7.0
dbt-semantic-interfaces==0.4.0
```

- **Fungsi:** Menyediakan daftar paket Python yang dibutuhkan untuk menjalankan dbt dengan koneksi PostgreSQL.

l. init.sql

File ini digunakan untuk menginisialisasi database dengan tabel mentah.

```
-- init.sql
CREATE TABLE raw.products (...);
CREATE TABLE raw.orders (...);
CREATE TABLE raw.order_details (...);
CREATE TABLE raw.brands (...);
```

- **Fungsi:** Menyediakan skrip SQL untuk membuat tabel mentah dalam database.

6. Menjalankan dbt

Jalankan perintah dbt berikut:

- `dbt deps`
  Mengunduh dan menginstal dependensi paket DBT yang didefinisikan dalam file `packages.yml`.

- `export DBT_PROFILES_DIR=$(pwd)/dbt-profiles`
  Perintah ini digunakan untuk memberitahu DBT di mana file `profiles.yml` berada jika tidak berada di direktori default (`~/.dbt/`). Ini berguna ketika Anda ingin menggunakan konfigurasi profil dari direktori tertentu dalam proyek Anda.

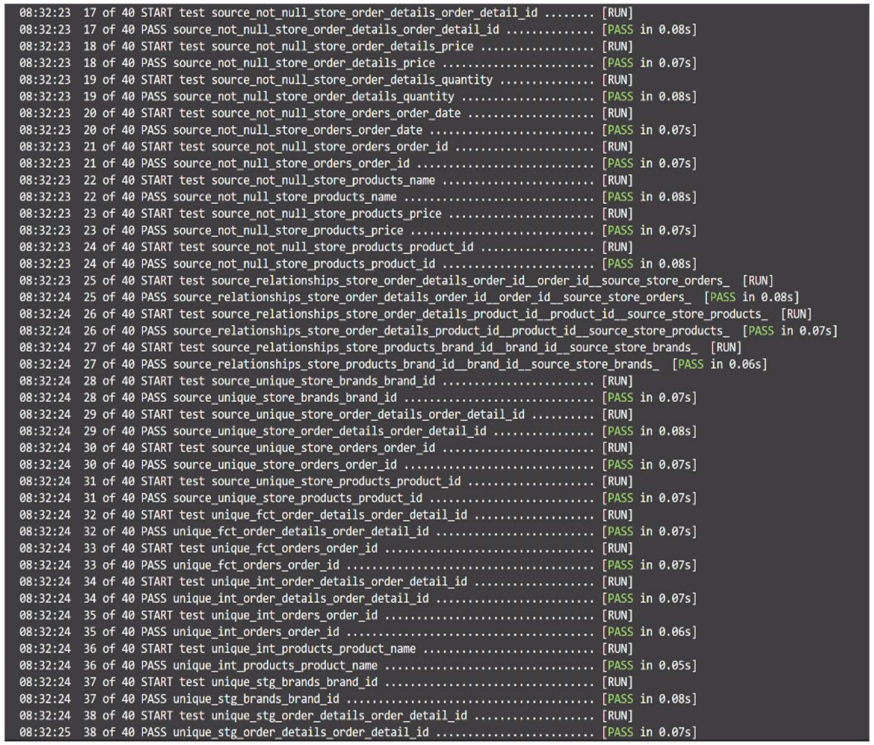
```
(.venv)
USER@Luna MINGW64 /c/Users/USER/Alta/belajar-bc/dbt/dbt-new (main)
$ export DBT_PROFILES_DIR=$(pwd)/dbt-profiles
```

- `dbt run`
  Mengeksekusi semua model DBT yang didefinisikan dalam proyek, mencakup semua model dalam direktori models dan mengupdate data dalam database sesuai dengan definisi model.

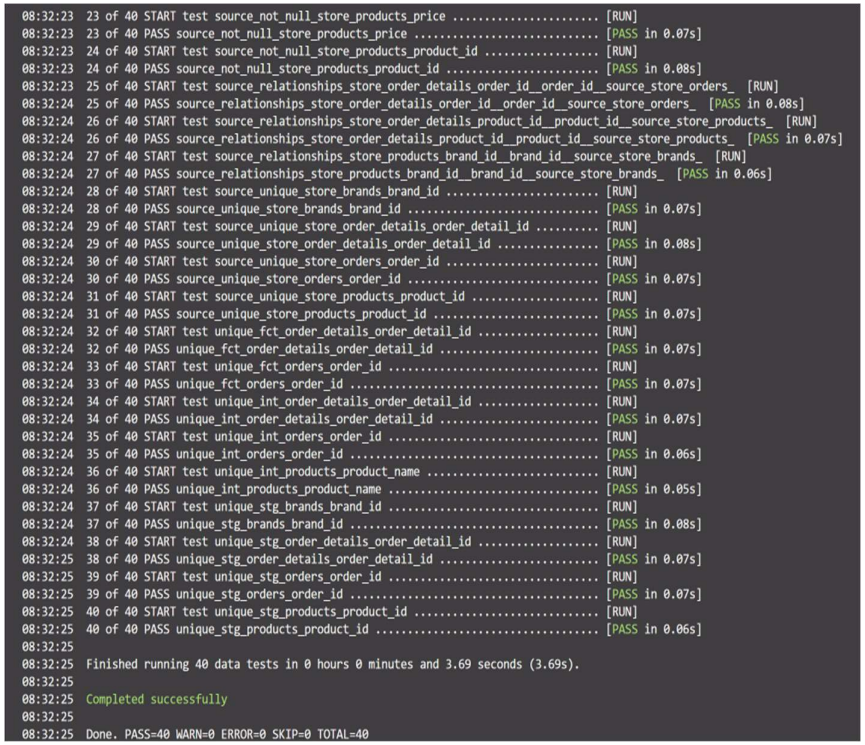
```
(.venv)
USER@Luna MINGW64 /c/Users/USER/Alta/belajar-bc/dbt/dbt-new/my_project (main)
$ dbt run
08:32:10  Running with dbt=1.8.4
08:32:11  Registered adapter: postgres=1.8.2
08:32:11  Found 13 models, 40 data tests, 4 sources, 802 macros
08:32:11
08:32:12  Concurrency: 1 threads (target='dev')
08:32:12
08:32:12  1 of 13 START sql view model public_stg.stg_brands ............................. [RUN]
08:32:12  1 of 13 OK created sql view model public_stg.stg_brands ........................ [CREATE VIEW in 0.20s]
08:32:12  2 of 13 START sql view model public_stg.stg_order_details ...................... [RUN]
08:32:12  2 of 13 OK created sql view model public_stg.stg_order_details ................. [CREATE VIEW in 0.11s]
08:32:12  3 of 13 START sql view model public_stg.stg_orders ............................. [RUN]
08:32:12  3 of 13 OK created sql view model public_stg.stg_orders ........................ [CREATE VIEW in 0.13s]
08:32:12  4 of 13 START sql view model public_stg.stg_products ........................... [RUN]
08:32:13  4 of 13 OK created sql view model public_stg.stg_products ...................... [CREATE VIEW in 0.10s]
08:32:13  5 of 13 START sql table model public_int.int_orders ............................ [RUN]
08:32:13  5 of 13 OK created sql table model public_int.int_orders ....................... [SELECT 391 in 0.17s]
08:32:13  6 of 13 START sql table model public_int.int_products .......................... [RUN]
08:32:13  6 of 13 OK created sql table model public_int.int_products ..................... [SELECT 4 in 0.14s]
08:32:13  7 of 13 START sql table model public_fact.fct_orders ........................... [RUN]
08:32:13  7 of 13 OK created sql table model public_fact.fct_orders ...................... [SELECT 391 in 0.14s]
08:32:13  8 of 13 START sql table model public_int.int_order_details ..................... [RUN]
08:32:13  8 of 13 OK created sql table model public_int.int_order_details ................ [SELECT 391 in 0.15s]
08:32:13  9 of 13 START sql table model public_mart.mart_finance_orders .................. [RUN]
08:32:13  9 of 13 OK created sql table model public_mart.mart_finance_orders ............. [SELECT 195 in 0.13s]
08:32:13  10 of 13 START sql table model public_mart.mart_marketing_orders ............... [RUN]
08:32:13  10 of 13 OK created sql table model public_mart.mart_marketing_orders .......... [SELECT 196 in 0.12s]
08:32:13  11 of 13 START sql table model public_fact.fct_daily_sales ..................... [RUN]
08:32:14  11 of 13 OK created sql table model public_fact.fct_daily_sales ................ [SELECT 391 in 0.11s]
08:32:14  12 of 13 START sql table model public_fact.fct_order_details ................... [RUN]
08:32:14  12 of 13 OK created sql table model public_fact.fct_order_details .............. [SELECT 391 in 0.12s]
08:32:14  13 of 13 START sql table model public_mart.mart_monthly_sales_summary .......... [RUN]
08:32:14  13 of 13 OK created sql table model public_mart.mart_monthly_sales_summary ..... [SELECT 8 in 0.13s]
08:32:14
08:32:14  Finished running 4 view models, 9 table models in 0 hours 0 minutes and 2.63 seconds (2.63s).
08:32:14
08:32:14  Completed successfully
08:32:14
08:32:14  Done. PASS=13 WARN=0 ERROR=0 SKIP=0 TOTAL=13
```

- `dbt test`
  menjalankan semua pengujian data yang didefinisikan dalam proyek Anda, baik pengujian bawaan DBT (seperti pengujian unik, tidak null, dan hubungan referensial) maupun pengujian kustom yang Anda definisikan dalam file `tests`.

```
(.venv)
USER@Luna MINGW64 /c/Users/USER/Alta/belajar-bc/dbt/dbt-new/my_project (main)
$ dbt test
08:32:20  Running with dbt=1.8.4
08:32:21  Registered adapter: postgres=1.8.2
08:32:21  Found 13 models, 40 data tests, 4 sources, 802 macros
08:32:21
08:32:21  Concurrency: 1 threads (target='dev')
08:32:21
08:32:21  1 of 40 START test dbt_expectations_expect_table_row_count_to_equal_other_table_fct_orders_ref_int_order_details_  [RUN]
08:32:22  1 of 40 PASS dbt_expectations_expect_table_row_count_to_equal_other_table_fct_orders_ref_int_order_details_  [PASS in 0.16s]
08:32:22  2 of 40 START test dbt_expectations_expect_table_row_count_to_equal_other_table_int_order_details_ref_stg_order_details_  [RUN]
08:32:22  2 of 40 PASS dbt_expectations_expect_table_row_count_to_equal_other_table_int_order_details_ref_stg_order_details_  [PASS in 0.08s]
08:32:22  3 of 40 START test dbt_utils_unique_combination_of_columns_fct_order_details_order_id__product_id  [RUN]
08:32:22  3 of 40 PASS dbt_utils_unique_combination_of_columns_fct_order_details_order_id__product_id  [PASS in 0.09s]
08:32:22  4 of 40 START test not_null_fct_order_details_order_detail_id ..... [RUN]
08:32:22  4 of 40 PASS not_null_fct_order_details_order_detail_id ..... [PASS in 0.12s]
08:32:22  5 of 40 START test not_null_fct_orders_order_id ..... [RUN]
08:32:22  5 of 40 PASS not_null_fct_orders_order_id ..... [PASS in 0.08s]
08:32:22  6 of 40 START test not_null_int_order_details_order_detail_id ..... [RUN]
08:32:22  6 of 40 PASS not_null_int_order_details_order_detail_id ..... [PASS in 0.07s]
08:32:22  7 of 40 START test not_null_int_orders_order_id ..... [RUN]
08:32:22  7 of 40 PASS not_null_int_orders_order_id ..... [PASS in 0.06s]
08:32:22  8 of 40 START test not_null_int_products_brand_name ..... [RUN]
08:32:22  8 of 40 PASS not_null_int_products_brand_name ..... [PASS in 0.06s]
08:32:22  9 of 40 START test not_null_int_products_product_name ..... [RUN]
08:32:22  9 of 40 PASS not_null_int_products_product_name ..... [PASS in 0.08s]
08:32:22  10 of 40 START test not_null_stg_brands_brand_id ..... [RUN]
08:32:22  10 of 40 PASS not_null_stg_brands_brand_id ..... [PASS in 0.08s]
08:32:22  11 of 40 START test not_null_stg_order_details_order_detail_id ..... [RUN]
08:32:22  11 of 40 PASS not_null_stg_order_details_order_detail_id ..... [PASS in 0.08s]
08:32:22  12 of 40 START test not_null_stg_orders_order_id ..... [RUN]
08:32:23  12 of 40 PASS not_null_stg_orders_order_id ..... [PASS in 0.07s]
08:32:23  13 of 40 START test not_null_stg_products_brand_id ..... [RUN]
08:32:23  13 of 40 PASS not_null_stg_products_brand_id ..... [PASS in 0.07s]
08:32:23  14 of 40 START test not_null_stg_products_product_id ..... [RUN]
08:32:23  14 of 40 PASS not_null_stg_products_product_id ..... [PASS in 0.09s]
08:32:23  15 of 40 START test source_not_null_store_brands_brand_id ..... [RUN]
08:32:23  15 of 40 PASS source_not_null_store_brands_brand_id ..... [PASS in 0.07s]
08:32:23  16 of 40 START test source_not_null_store_brands_name ..... [RUN]
08:32:23  16 of 40 PASS source_not_null_store_brands_name ..... [PASS in 0.08s]
08:32:23  17 of 40 START test source_not_null_store_order_details_order_detail_id ..... [RUN]
08:32:23  17 of 40 PASS source_not_null_store_order_details_order_detail_id ..... [PASS in 0.08s]
08:32:23  18 of 40 START test source_not_null_store_order_details_price ..... [RUN]
```



```
08:32:23  17 of 40 START test source_not_null_store_order_details_order_detail_id ..... [RUN]
08:32:23  17 of 40 PASS source_not_null_store_order_details_order_detail_id ..... [PASS in 0.08s]
08:32:23  18 of 40 START test source_not_null_store_order_details_price ..... [RUN]
08:32:23  18 of 40 PASS source_not_null_store_order_details_price ..... [PASS in 0.07s]
08:32:23  19 of 40 START test source_not_null_store_order_details_quantity ..... [RUN]
08:32:23  19 of 40 PASS source_not_null_store_order_details_quantity ..... [PASS in 0.08s]
08:32:23  20 of 40 START test source_not_null_store_orders_order_date ..... [RUN]
08:32:23  20 of 40 PASS source_not_null_store_orders_order_date ..... [PASS in 0.07s]
08:32:23  21 of 40 START test source_not_null_store_orders_order_id ..... [RUN]
08:32:23  21 of 40 PASS source_not_null_store_orders_order_id ..... [PASS in 0.07s]
08:32:23  22 of 40 START test source_not_null_store_products_name ..... [RUN]
08:32:23  22 of 40 PASS source_not_null_store_products_name ..... [PASS in 0.08s]
08:32:23  23 of 40 START test source_not_null_store_products_price ..... [RUN]
08:32:23  23 of 40 PASS source_not_null_store_products_price ..... [PASS in 0.07s]
08:32:23  24 of 40 START test source_not_null_store_products_product_id ..... [RUN]
08:32:23  24 of 40 PASS source_not_null_store_products_product_id ..... [PASS in 0.08s]
08:32:23  25 of 40 START test source_relationships_store_order_details_order_id__order_id__source_store_orders_ [RUN]
08:32:24  25 of 40 PASS source_relationships_store_order_details_order_id__order_id__source_store_orders_ [PASS in 0.08s]
08:32:24  26 of 40 START test source_relationships_store_order_details_product_id__product_id__source_store_products_ [RUN]
08:32:24  26 of 40 PASS source_relationships_store_order_details_product_id__product_id__source_store_products_ [PASS in 0.07s]
08:32:24  27 of 40 START test source_relationships_store_products_brand_id__brand_id__source_store_brands_ [RUN]
08:32:24  27 of 40 PASS source_relationships_store_products_brand_id__brand_id__source_store_brands_ [PASS in 0.06s]
08:32:24  28 of 40 START test source_unique_store_brands_brand_id ..... [RUN]
08:32:24  28 of 40 PASS source_unique_store_brands_brand_id ..... [PASS in 0.07s]
08:32:24  29 of 40 START test source_unique_store_order_details_order_detail_id ..... [RUN]
08:32:24  29 of 40 PASS source_unique_store_order_details_order_detail_id ..... [PASS in 0.08s]
08:32:24  30 of 40 START test source_unique_store_orders_order_id ..... [RUN]
08:32:24  30 of 40 PASS source_unique_store_orders_order_id ..... [PASS in 0.07s]
08:32:24  31 of 40 START test source_unique_store_products_product_id ..... [RUN]
08:32:24  31 of 40 PASS source_unique_store_products_product_id ..... [PASS in 0.07s]
08:32:24  32 of 40 START test unique_fct_order_details_order_detail_id ..... [RUN]
08:32:24  32 of 40 PASS unique_fct_order_details_order_detail_id ..... [PASS in 0.07s]
08:32:24  33 of 40 START test unique_fct_orders_order_id ..... [RUN]
08:32:24  33 of 40 PASS unique_fct_orders_order_id ..... [PASS in 0.07s]
08:32:24  34 of 40 START test unique_int_order_details_order_detail_id ..... [RUN]
08:32:24  34 of 40 PASS unique_int_order_details_order_detail_id ..... [PASS in 0.07s]
08:32:24  35 of 40 START test unique_int_orders_order_id ..... [RUN]
08:32:24  35 of 40 PASS unique_int_orders_order_id ..... [PASS in 0.06s]
08:32:24  36 of 40 START test unique_int_products_product_name ..... [RUN]
08:32:24  36 of 40 PASS unique_int_products_product_name ..... [PASS in 0.05s]
08:32:24  37 of 40 START test unique_stg_brands_brand_id ..... [RUN]
08:32:24  37 of 40 PASS unique_stg_brands_brand_id ..... [PASS in 0.08s]
08:32:24  38 of 40 START test unique_stg_order_details_order_detail_id ..... [RUN]
08:32:25  38 of 40 PASS unique_stg_order_details_order_detail_id ..... [PASS in 0.07s]
```



```
08:32:23  23 of 40 START test source_not_null_store_products_price ..... [RUN]
08:32:23  23 of 40 PASS source_not_null_store_products_price ..... [PASS in 0.07s]
08:32:23  24 of 40 START test source_not_null_store_products_product_id ..... [RUN]
08:32:23  24 of 40 PASS source_not_null_store_products_product_id ..... [PASS in 0.08s]
08:32:23  25 of 40 START test source_relationships_store_order_details_order_id__order_id__source_store_orders_ [RUN]
08:32:24  25 of 40 PASS source_relationships_store_order_details_order_id__order_id__source_store_orders_ [PASS in 0.08s]
08:32:24  26 of 40 START test source_relationships_store_order_details_product_id__product_id__source_store_products_ [RUN]
08:32:24  26 of 40 PASS source_relationships_store_order_details_product_id__product_id__source_store_products_ [PASS in 0.07s]
08:32:24  27 of 40 START test source_relationships_store_products_brand_id__brand_id__source_store_brands_ [RUN]
08:32:24  27 of 40 PASS source_relationships_store_products_brand_id__brand_id__source_store_brands_ [PASS in 0.06s]
08:32:24  28 of 40 START test source_unique_store_brands_brand_id ..... [RUN]
08:32:24  28 of 40 PASS source_unique_store_brands_brand_id ..... [PASS in 0.07s]
08:32:24  29 of 40 START test source_unique_store_order_details_order_detail_id ..... [RUN]
08:32:24  29 of 40 PASS source_unique_store_order_details_order_detail_id ..... [PASS in 0.08s]
08:32:24  30 of 40 START test source_unique_store_orders_order_id ..... [RUN]
08:32:24  30 of 40 PASS source_unique_store_orders_order_id ..... [PASS in 0.07s]
08:32:24  31 of 40 START test source_unique_store_products_product_id ..... [RUN]
08:32:24  31 of 40 PASS source_unique_store_products_product_id ..... [PASS in 0.07s]
08:32:24  32 of 40 START test unique_fct_order_details_order_detail_id ..... [RUN]
08:32:24  32 of 40 PASS unique_fct_order_details_order_detail_id ..... [PASS in 0.07s]
08:32:24  33 of 40 START test unique_fct_orders_order_id ..... [RUN]
08:32:24  33 of 40 PASS unique_fct_orders_order_id ..... [PASS in 0.07s]
08:32:24  34 of 40 START test unique_int_order_details_order_detail_id ..... [RUN]
08:32:24  34 of 40 PASS unique_int_order_details_order_detail_id ..... [PASS in 0.07s]
08:32:24  35 of 40 START test unique_int_orders_order_id ..... [RUN]
08:32:24  35 of 40 PASS unique_int_orders_order_id ..... [PASS in 0.06s]
08:32:24  36 of 40 START test unique_int_products_product_name ..... [RUN]
08:32:24  36 of 40 PASS unique_int_products_product_name ..... [PASS in 0.05s]
08:32:24  37 of 40 START test unique_stg_brands_brand_id ..... [RUN]
08:32:24  37 of 40 PASS unique_stg_brands_brand_id ..... [PASS in 0.08s]
08:32:24  38 of 40 START test unique_stg_order_details_order_detail_id ..... [RUN]
08:32:25  38 of 40 PASS unique_stg_order_details_order_detail_id ..... [PASS in 0.07s]
08:32:25  39 of 40 START test unique_stg_orders_order_id ..... [RUN]
08:32:25  39 of 40 PASS unique_stg_orders_order_id ..... [PASS in 0.07s]
08:32:25  40 of 40 START test unique_stg_products_product_id ..... [RUN]
08:32:25  40 of 40 PASS unique_stg_products_product_id ..... [PASS in 0.06s]
08:32:25
08:32:25  Finished running 40 data tests in 0 hours 0 minutes and 3.69 seconds (3.69s).
08:32:25
08:32:25  Completed successfully
08:32:25
08:32:25  Done. PASS=40 WARN=0 ERROR=0 SKIP=0 TOTAL=40
```

- `dbt docs generate`
  Perintah ini menghasilkan dokumentasi HTML untuk proyek DBT Anda berdasarkan model, sumber, dan pengujian yang didefinisikan. Dokumentasi ini mencakup diagram lineage, definisi model, dan informasi pengujian.

```
(.venv)
USER@Luna MINGW64 /c/Users/USER/Alta/belajar-bc/dbt/dbt-new/my_project (main)
$ dbt docs generate
08:32:34  Running with dbt=1.8.4
08:32:34  Registered adapter: postgres=1.8.2
08:32:34  Found 13 models, 40 data tests, 4 sources, 802 macros
08:32:34
08:32:35  Concurrency: 1 threads (target='dev')
08:32:35
08:32:36  Building catalog
08:32:36  Catalog written to C:\Users\USER\Alta\belajar-bc\dbt\dbt-new\my_project\target\catalog.json
```

- `dbt docs serve`
  Menyajikan dokumentasi DBT yang dihasilkan pada server lokal.

```
(.venv)
USER@Luna MINGW64 /c/Users/USER/Alta/belajar-bc/dbt/dbt-new/my_project (main)
$ dbt docs serve
08:32:58  Running with dbt=1.8.4
Serving docs at 8080
To access from your browser, navigate to: http://localhost:8080



Press Ctrl+C to exit.
127.0.0.1 - - [07/Aug/2024 15:32:59] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [07/Aug/2024 15:33:00] "GET /manifest.json?cb=1723019580031 HTTP/1.1" 200 -
127.0.0.1 - - [07/Aug/2024 15:33:00] "GET /catalog.json?cb=1723019580031 HTTP/1.1" 200 -
127.0.0.1 - - [07/Aug/2024 15:33:00] code 404, message File not found
127.0.0.1 - - [07/Aug/2024 15:33:00] "GET /$%7Brequire('./assets/favicons/favicon.ico')%7D HTTP/1.1" 404 -
```

Tampilan dari dbt serve: