Nama : Al Fitra Nur Ramadhani

NIM : 202210370311264

Mata Kuliah : Pemodelan dan Simulasi Data B

**Laporan Master Data Warehousing, Dimensional Modeling & ETL Process Sales Data**

1. **Deskripsi**

Laporan ini mendokumentasikan implementasi *Master Data Warehousing* dengan pendekatan *Dimensional Modeling* dan proses *Extract, Transform, Load* (ETL) untuk data penjualan. Data sumber berasal dari dua file CSV, yaitu Fact\_Sales\_1.csv dan Fact\_Sales\_2.csv, yang diimpor ke tabel public.sales. Proses ETL mengubah data mentah ini menjadi struktur *data warehouse* yang terdiri dari tabel fakta (core.sales) dan tabel dimensi (core.dim\_payment, core.dim\_product). Selain itu, proses *incremental load* diterapkan untuk efisiensi pembaruan data. Tujuan utama adalah menyediakan data yang terstruktur untuk analisis bisnis, seperti profitabilitas, pola pembayaran, dan penggunaan kartu loyalitas.

Proses ini menggunakan *Docker* untuk mengelola lingkungan, *Python* untuk logika ETL, *PostgreSQL* sebagai DBMS, *PGAdmin 4* untuk manajemen basis data, dan *Visual Studio Code* untuk pengembangan skrip. Laporan ini akan menjelaskan setiap langkah secara rinci, termasuk kondisi *before* dan *after*, sebab-akibat, serta dampak dari pengimporan kedua file CSV.

1. **Link GitHub =**
2. **Tools Yang Digunakan**

- **Docker**: Mengelola kontainer untuk *PostgreSQL*, *PGAdmin 4*, dan aplikasi *Python* ETL melalui docker-compose.yml.

- **Python 3.11**: Digunakan untuk menjalankan skrip ETL (etl\_sales.py untuk *full load* dan etl\_sales\_incremental.py untuk *incremental load*). Library psycopg2 menghubungkan *Python* dengan *PostgreSQL*.

- **PostgreSQL**: DBMS untuk menyimpan data mentah (public.sales), data sementara (staging.sales), dan *data warehouse* (core.sales, core.dim\_payment, core.dim\_product).

- **PGAdmin 4**: Antarmuka untuk mengelola basis data, menjalankan query SQL, dan memeriksa struktur tabel.

- **Visual Studio Code**: IDE untuk mengembangkan skrip *Python*, mengelola file konfigurasi, dan menjalankan perintah terminal.

1. **Implementasi Step by Step**

**1. Persiapan Lingkungan**

* Konfigurasi Docker:
  + File docker-compose.yml mendefinisikan tiga layanan:
    - postgres: Menjalankan *PostgreSQL* dengan database mydb, pengguna postgres, dan kata sandi \*\*\*\*\*\*\*\*\*. Data disimpan di volume postgres-data.
    - pgadmin: Menyediakan antarmuka *PGAdmin 4* dengan email alfitranurr@gmail.com dan kata sandi yang sama.
    - python-etl: Menjalankan skrip ETL menggunakan image *Python 3.11*, dengan dependensi diinstall dari requirements.txt.
* Command untuk menjalankan layanan di VSCode:

‘docker-compose up postgres pgadmin’

‘docker-compose up python-etl’

* Persiapan Data:

File CSV (Fact\_Sales\_1.csv) dan (Fact\_Sales\_2.csv) diimpor ke tabel public.sales menggunakan perintah :

\copy public.sales (transaction\_id, transactional\_date, product\_id, customer\_id, payment, credit\_card, loyalty\_card, cost, quantity, price) FROM '/data/Fact\_Sales\_2.csv' DELIMITER ',' CSV HEADER;

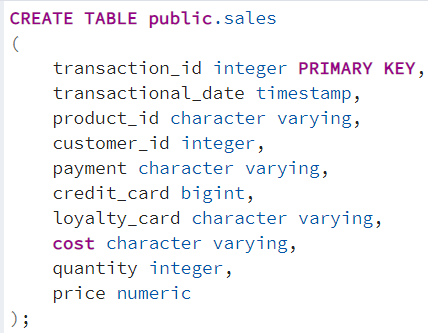
**2. Pembuatan Struktur Database**

**Skema dan Tabel:**

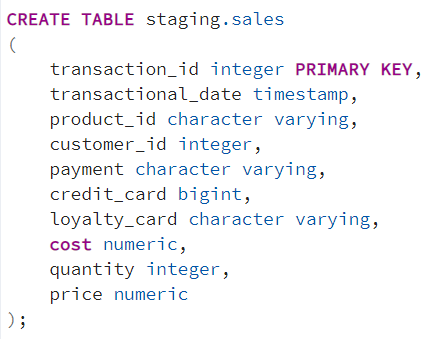
* Skema public, staging, dan core dibuat di *PostgreSQL :*

****

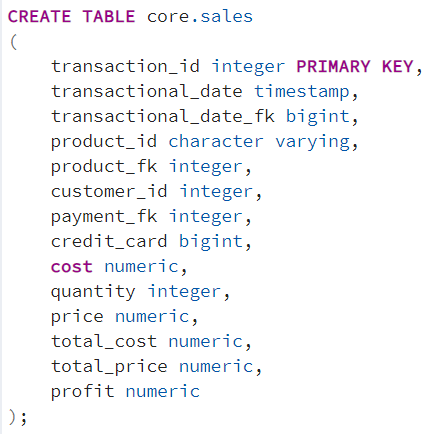
* Tabel public.sales menyimpan data mentah dengan struktur:



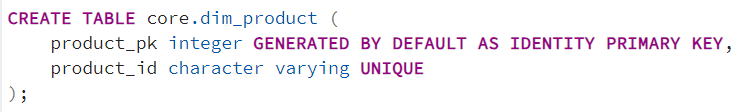
* Tabel staging.sales untuk data sementara dengan tipe data yang telah disesuaikan:

****

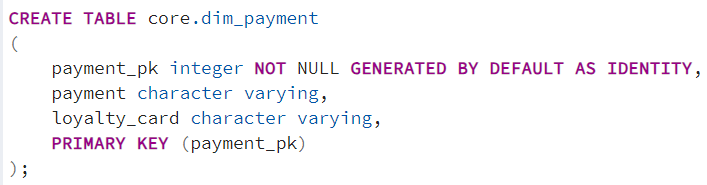
* Tabel fakta core.sales dengan kolom tambahan untuk metrik seperti total\_cost, total\_price, dan profit:



* Tabel dimensi core.dim\_product untuk menyimpan ID produk unik:



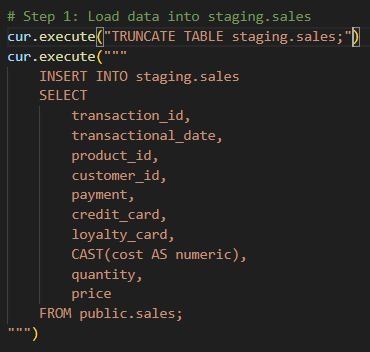
* Tabel dimensi core.dim\_payment untuk menyimpan kombinasi metode pembayaran dan status kartu loyalitas:



**3. Proses ETL (Full Load) - etl\_sales.py**

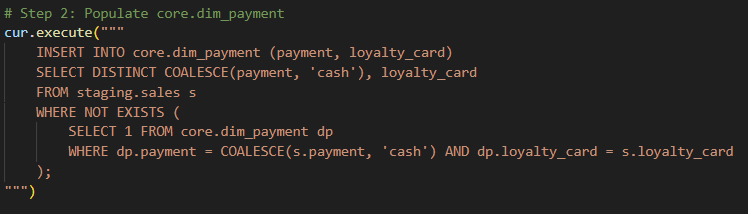
**Extract:**

* Data dari public.sales dimuat ke staging.sales setelah konversi tipe data (misalnya, cost dari *string* ke *numeric*):

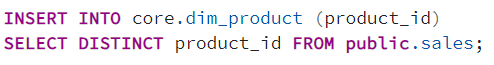


**Transform:**

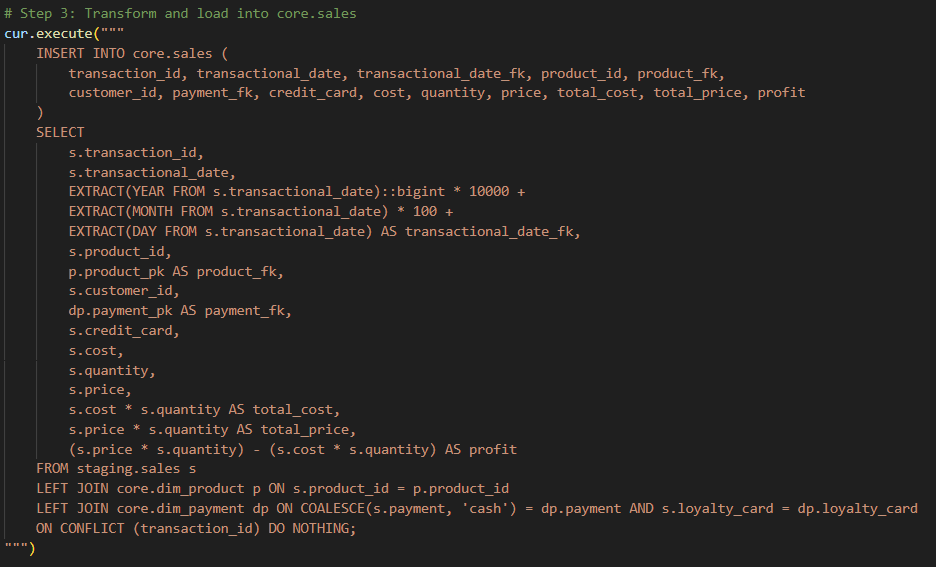
* Memastikan nilai payment yang kosong diisi dengan 'cash' menggunakan COALESCE.
* Mengisi tabel dimensi core.dim\_payment dengan kombinasi unik metode pembayaran dan kartu loyalitas:



* Mengisi tabel dimensi core.dim\_product dengan ID produk unik:

****

* Transformasi data ke core.sales, termasuk perhitungan total\_cost, total\_price, dan profit:

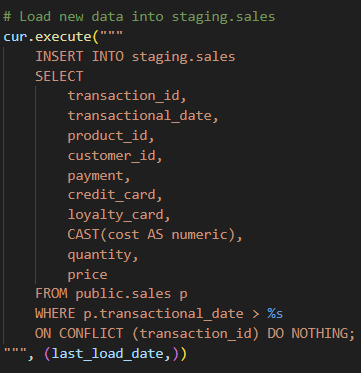
****

**Load:**

* Data yang telah ditransformasi dimuat ke tabel fakta core.sales dengan kunci asing (product\_fk, payment\_fk) yang sesuai.

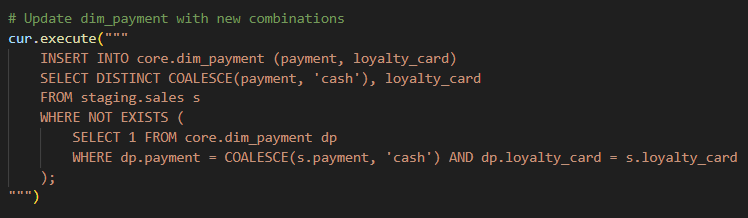
**4. Proses ETL (Incremental Load) - etl\_sales\_incremental.py**

**Extract**:

* Memuat hanya data baru berdasarkan last\_load\_date dari tabel etl\_metadata:

**Transform:**

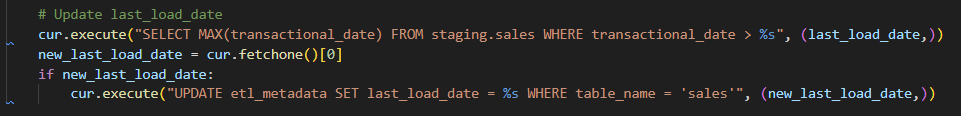
* Memperbarui core.dim\_payment dengan kombinasi baru:

****

* Transformasi data baru ke core.sales dengan logika yang sama seperti full load, tetapi hanya untuk data dengan transactional\_date > last\_load\_date.

**Load:**

* Memuat data baru ke core.sales dan memperbarui last\_load\_date di etl\_metadata:

****

**5. Verifikasi Data**

**Memeriksa Tabel Dimensi:**

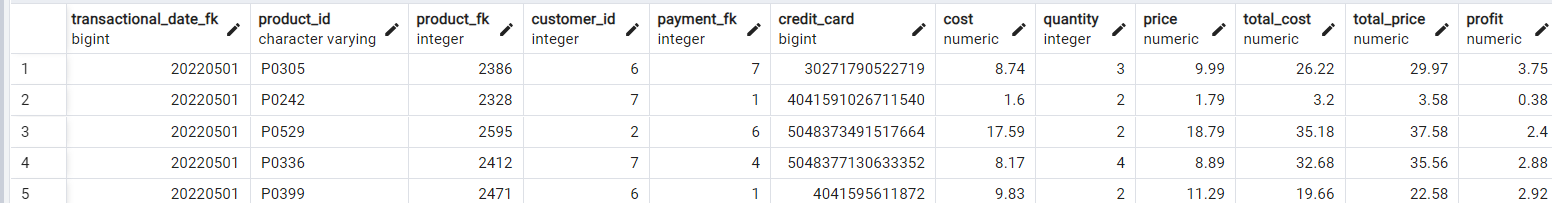
* Query SELECT \* FROM core.dim\_payment; digunakan untuk memverifikasi isi tabel core.dim\_payment. Contoh hasil berdasarkan dim\_payment.csv:

****

Penjelasan: Tabel ini berisi kombinasi unik metode pembayaran dan status kartu loyalitas. Kolom payment\_pk adalah kunci utama yang dihasilkan secara otomatis, digunakan sebagai kunci asing di core.sales.

**Memeriksa Tabel Fakta**:

* Query SELECT \* FROM core.sales WHERE transactional\_date >= '2022-05-01' LIMIT 5; digunakan untuk memeriksa data di core.sales. Contoh hasil setelah proses ETL:



Penjelasan: Tabel core.sales berisi data transaksi yang telah ditransformasi, dengan kolom seperti transactional\_date\_fk (format YYYYMMDD untuk analisis waktu), product\_fk (kunci asing ke core.dim\_product), payment\_fk (kunci asing ke core.dim\_payment), serta metrik total\_cost, total\_price, dan profit.

1. **Kesimpulan**

- Proses ETL berhasil mengubah data mentah dari public.sales menjadi struktur *data warehouse* yang terdiri dari tabel fakta (core.sales) dan tabel dimensi (core.dim\_payment, core.dim\_product).

- Pendekatan *dimensional modeling* memungkinkan analisis data yang efisien dengan pemisahan data transaksional dan atribut (metode pembayaran, produk).

- Proses *incremental load* meningkatkan efisiensi dengan hanya memproses data baru, cocok untuk pembaruan data secara berkala.

- Penggunaan *Docker* memastikan lingkungan yang konsisten dan portabel, sementara *PostgreSQL* dan *Python* mendukung skalabilitas dan fleksibilitas proses ETL.