Perancangan Desain Logikal dan Fisikal untuk Industri Manufaktur : Tesla

Disusun oleh kelompok 7 DW RA

Elia Meylani Simanjuntak (122450026)
Program Studi Sains Data, Fakultas Sains

Sahid Maulana (122450109)
Program Studi Sains Data, Fakultas Sains

Chalifia Wananda (122450076)
Program Studi Sains Data, Fakultas Sains

Muhammad Rafif Vivaldi (122140026)
Program Studi Teknik Informatika, Fakultas Teknologi Industri

INSTITUT TEKNOLOGI SUMATERA
LAMPUNG SELATAN
2025

1. Desain Konseptual

1.1 Desain Tabel Dimensi

a. Skema

Pada proyek ini, kami mengadopsi Star Schema sebagai desain konseptual utama untuk data warehouse. Dalam skema ini, satu tabel fakta berada di pusat dan dikelilingi oleh beberapa tabel dimensi yang langsung terhubung melalui foreign key. Adapun beberapa alasan dan pertimbangan Star Schema dipilih diantaranya :

- Sederhana dan intuitif, cocok untuk pengguna teknis maupun non-teknis.
- Optimal untuk kueri OLAP, seperti agregasi, slicing, dicing, drill-down.
- Mendukung ekspansi, karena dimensi bisa ditambah tanpa mengubah struktur inti.
- Kinerja kueri cepat pada data yang besar karena join minimal (langsung ke tabel fakta).

b. Identifikasi Dimensi Berdasarkan Kebutuhan Analitik

Berdasarkan studi kasus Tesla dan asumsi kebutuhan bisnis, kami mengidentifikasi 5 entitas utama yang akan menjadi tabel dimensi. Dimensi ini merepresentasikan atribut deskriptif yang membantu menjelaskan data kuantitatif pada tabel fakta.

Nama Dimensi	Atribut Utama	Tujuan Analitik
Dim_Waktu	date, month, quarter, year	Melacak tren waktu, musiman, tahunan, siklus produksi
Dim_Model	model, battery_type, drive_type, color	Analisis produksi: preferensi pelanggan, performa per model
Dim_Wilayah	city, region, country	Distribusi kendaraan berdasarkan lokasi, geografis
Dim_Performa	software_version, efficiency, charging_time, autopilot_enabled	Menilai dampak teknologi terhadap efisiensi dan fitur kendaraan
Dim_Pelanggan	warranty_period, update_frequency, customer_rating	Segmentasi dan loyalitas pelanggan berdasarkan karakteristik pengguna

c. Implementasi Tabel Dimensi

Berikut adalah rancangan struktur masing-masing tabel dimensi menggunakan SQL. Setiap perintah CREATE TABLE dirancang agar siap menerima data saat tersedia.

- Dim_Waktu

  Digunakan untuk mendukung analisis berdasarkan waktu (tanggal, bulan, tahun).

```sql
CREATE TABLE Dim_Waktu (
    waktu_id SERIAL PRIMARY KEY,
    date DATE,              -- Tanggal spesifik
    month INT,              -- Bulan dalam bentuk angka
    quarter INT,            -- Kuartal (1 hingga 4)
    year INT                -- Tahun
);
```

Deskripsi: Tabel ini menyimpan informasi kalender yang digunakan untuk agregasi dan analisis waktu, seperti tren tahunan dan musiman.

- Dim_Model

  Merepresentasikan spesifikasi dari unit kendaraan seperti model dan fitur teknisnya.

```sql
CREATE TABLE Dim_Model (
    model_id SERIAL PRIMARY KEY,
    model TEXT,             -- Nama model kendaraan (misal: Model 3, Model Y)
    battery_type TEXT,      -- Jenis baterai (Long Range, Performance, dll.)
    drive_type TEXT,        -- Tipe penggerak (AWD, RWD)
    color TEXT              -- Warna kendaraan
);
```

Deskripsi: Tabel ini berisi deskripsi spesifikasi kendaraan Tesla yang digunakan untuk analisis performa dan preferensi pelanggan.

- Dim_Wilayah

  Untuk mendukung analisis distribusi kendaraan berdasarkan lokasi geografis.

```sql
CREATE TABLE Dim_Wilayah (
    wilayah_id SERIAL PRIMARY KEY,
    city TEXT,              -- Kota tujuan distribusi
    region TEXT,            -- Wilayah atau provinsi
    country TEXT            -- Negara
);
```

Deskripsi: Tabel ini menyimpan informasi geografis untuk mendukung pelacakan distribusi kendaraan dan analisis regional.

- Dim_Performa

Menampung atribut teknis kendaraan yang memengaruhi pengalaman dan efisiensi penggunaan.

```
1. CREATE TABLE Dim_Performa (
2.    performa_id SERIAL PRIMARY KEY,
3.    software_version TEXT,      -- Versi
   perangkat lunak terinstal
4.    efficiency FLOAT,           -- Efisiensi
   energi kendaraan (Wh/km)
5.    charging_time INT,          -- Waktu
   pengisian daya rata-rata (menit)
6.    autopilot_enabled BOOLEAN  -- Status
   aktif/tidaknya fitur Autopilot
7. );
```

Deskripsi: Tabel ini mencatat fitur teknis kendaraan untuk memungkinkan evaluasi dampak inovasi terhadap efisiensi dan kenyamanan.

- Dim_Pelanggan

Dimensi ini mendukung segmentasi pelanggan berdasarkan atribut layanan dan kepuasan.

```
1. CREATE TABLE Dim_Pelanggan (
2.    pelanggan_id SERIAL PRIMARY KEY,
3.    warranty_period INT,        -- Durasi
   garansi (bulan)
4.    update_frequency INT,       -- Frekuensi
   pembaruan software (bulan)
5.    customer_rating FLOAT       -- Rating
   pelanggan terhadap kendaraan (skala 1-5)
6. );
```

Deskripsi: Tabel ini merepresentasikan atribut pelanggan yang digunakan untuk segmentasi loyalitas dan kepuasan.

1.2 Desain Tabel Fakta

a. Peran dan Fungsi Tabel Fakta

Tabel fakta merupakan komponen utama dalam arsitektur Star Schema. Ia berfungsi sebagai pusat penyimpanan data kuantitatif atau matrix yang mencerminkan kinerja operasional dan bisnis. Dalam konteks studi kasus Tesla, tabel fakta dirancang untuk merekam metrik pengiriman kendaraan yang dapat dianalisis dari berbagai sudut pandang dimensi.

b. Rancangan Tabel Fakta

- Fakta_Pengiriman_Kendaraan

Tabel ini akan mencatat data pengiriman kendaraan berdasarkan model, wilayah, waktu, performa teknis, dan profil pelanggan. Karena data input belum tersedia, struktur dirancang berdasarkan estimasi kebutuhan bisnis.

Nama Kolom	Tipe Data	Deskripsi
delivery_id	SERIAL	Primary key, identifikasi unik setiap entri pengiriman
model_id	INT	Foreign key ke Dim_Model
wilayah_id	INT	Foreign key ke Dim_Wilayah
waktu_id	INT	Foreign key ke Dim_Waktu
performa_id	INT	Foreign key ke Dim_Performa
pelanggan_id	INT	Foreign key ke Dim_Pelanggan
units_delievered	INT	Jumlah kendaraan yang dikirim
production_delay_days	INT	Lama kendaraan produksi (dalam hari)
average_price	NUMERIC	Harga rata - rata per kendaraan (USD)
service_visits	INT	Jumlah kunjungan servis kendaraan
battery_replacement_rate	FLOAT	Rasio penggantian baterai
total_miles_driven	INT	Total jarak tempuh kendaraan (dalam mil)
number_of_recalls	INT	Jumlah recall produk
CO2_saved	FLOAT	Estimasi pengurangan emisi CO2 (dalam ton)
resale_value	FLOAT	Persentase nilai jual kembali kendaraan
avg_customer_rating	FLOAT	Rata - rata penilaian pelanggan terhadap kendaraan

delievery_date	DATE	Tanggal Pengiriman kendaraan

c. Implementasi

Berikut adalah rancangan struktur tabel fakta dalam bentuk SQL. Tabel ini akan menjadi pusat integrasi data kuantitatif yang dapat dihubungkan ke semua tabel dimensi.

```
1. CREATE TABLE Fakta_Pengiriman_Kendaraan (
2.     delivery_id SERIAL PRIMARY KEY,
3.     model_id INT REFERENCES Dim_Model(model_id),
4.     wilayah_id INT REFERENCES
   Dim_Wilayah(wilayah_id),
5.     waktu_id INT REFERENCES Dim_Waktu(waktu_id),
6.     performa_id INT REFERENCES
   Dim_Performa(performa_id),
7.     pelanggan_id INT REFERENCES
   Dim_Pelanggan(pelanggan_id),
8.     units_delivered INT,                    -- Jumlah
   kendaraan dikirim
9.     production_delay_days INT,             --
   Keterlambatan produksi (hari)
10.      average_price NUMERIC,                --
   Harga rata-rata per kendaraan (USD)
11.      service_visits INT,                   --
   Jumlah servis kendaraan
12.      battery_replacement_rate FLOAT,       --
   Rasio penggantian baterai
13.      total_miles_driven INT,               --
   Total jarak tempuh (mil)
14.      number_of_recalls INT,                --
   Total recall kendaraan
15.      CO2_saved FLOAT,                      --
   Estimasi CO2 dihemat (ton)
16.      resale_value FLOAT,                   --
   Nilai jual kembali kendaraan (%)
17.      avg_customer_rating FLOAT,            --
   Rata-rata rating dari pelanggan
18.      delivery_date DATE                    --
   Tanggal pengiriman
19.   );
```

Tabel ini menjadi pusat data metrik operasional Tesla yang akan dianalisis berdasarkan dimensi waktu, produk, wilayah, performa teknis, dan pelanggan.

2. Desain Logikal dan Fisikal

2.1 Design and Implement Indeks

a. Tujuan dan Peran Indeks

Indeks merupakan komponen penting dalam arsitektur data warehouse karena memungkinkan peningkatan performa sistem, terutama dalam pengolahan query kompleks pada volume data yang besar. Indeks dirancang untuk mempercepat operasi pencarian, penyaringan, dan pengurutan data, serta mempercepat proses join antara tabel fakta dan tabel dimensi.

Dalam konteks data warehouse Tesla, indeks membantu mempercepat pengambilan data analitis seperti laporan distribusi kendaraan per wilayah, analisis performa berdasarkan model, dan evaluasi tren berdasarkan waktu.

b. Jenis-jenis dan Strategi Implementasi Indeks

1. Clustered Index

   Clustered index menyimpan data secara fisik di disk berdasarkan urutan kolom yang diindeks. Hanya satu clustered index yang bisa diterapkan pada satu tabel.

```
1. CREATE CLUSTERED INDEX idx_delivery_date
2. ON Fakta_Pengiriman_Kendaraan
   (delivery_date);
```

   Indeks ini meningkatkan performa query yang mengurutkan atau memfilter berdasarkan tanggal pengiriman kendaraan.

2. Non-Clustered Index

   Non-clustered index menyimpan pointer ke lokasi data asli. Beberapa non-clustered index dapat dibuat untuk satu tabel, dan sangat berguna untuk kolom yang sering digunakan dalam operasi JOIN, WHERE, atau GROUP BY.

```
1. CREATE INDEX idx_model_id
2. ON Fakta_Pengiriman_Kendaraan (model_id);
3.
4. CREATE INDEX idx_region_id
5. ON Fakta_Pengiriman_Kendaraan (wilayah_id);
6.
7. CREATE INDEX idx_customer_id
8. ON Fakta_Pengiriman_Kendaraan
   (pelanggan_id);
```

   Penjelasan :

3. Composite Index

Jika query sering mengakses kombinasi kolom secara bersamaan, maka composite index dapat meningkatkan efisiensi.

```
1. CREATE INDEX idx_model_region
2. ON Fakta_Pengiriman_Kendaraan (model_id,
   wilayah_id);
```

Berguna untuk query yang memfilter berdasarkan kombinasi model kendaraan dan wilayah distribusi.

2.2 Desain Storage

Desain penyimpanan dalam proyek ini mengikuti *Three-Tier Architecture* yang umum digunakan dalam pengembangan sistem data warehouse. Pendekatan ini memastikan data diproses secara terstruktur dan efisien, dari sumber mentah hingga penyajian analisis. Berikut adalah tiga lapisan yang diimplementasikan:

a. Tier 1: Data Source - Collection Layer

Lapisan ini bertugas untuk mengoleksi dan menyimpan data mentah dari berbagai sumber (misalnya CSV, API, database operasional). Data belum dibersihkan atau distandarisasi, dan digunakan sebagai bahan mentah untuk proses ETL.

```
1. CREATE TABLE Staging_Pengiriman (
2.     raw_id SERIAL PRIMARY KEY,
3.     model TEXT,
4.     battery_type TEXT,
5.     city TEXT,
6.     delivery_date TEXT,
7.     unit_delivered TEXT,
8.     customer_rating TEXT
9. );
```

Tabel Staging_Pengiriman menyimpan data mentah yang diekstrak dari sumber luar. Tipe data masih umum (TEXT) karena belum dibersihkan. Digunakan sementara sebelum ETL.

b. Tier 2: Warehouse Layer (Staging > ETL > Storage)

Lapisan tengah ini merupakan inti dari data warehouse. Data dari staging dibersihkan, ditransformasi, lalu dimasukkan ke dalam tabel dimensi dan fakta yang dirancang dalam Star Schema. Di sini juga dilakukan pengaturan partisi, indeks, dan alokasi filegroup.

- Pembuatan Filegroup untuk Optimasi Penyimpanan

```
1. ALTER DATABASE TeslaDW ADD FILEGROUP DW_Fakta;
2.
3. ALTER DATABASE TeslaDW
4. ADD FILE (
5.     NAME = DW_Fakta_Data,
6.     FILENAME = 'D:\SQLData\DW_Fakta.ndf',
7.     SIZE = 100MB,
```

```
    MAXSIZE = 1GB,
    FILEGROWTH = 50MB
) TO FILEGROUP DW_Fakta;
```

Filegroup DW_Fakta dipisahkan dari PRIMARY untuk memisahkan penyimpanan antara tabel dimensi dan fakta. Hal ini mempermudah manajemen ruang dan meningkatkan performa I/O.

- Kompresi Tabel Fakta

```
ALTER TABLE Fakta_Pengiriman_Kendaraan
REBUILD PARTITION = ALL
WITH (DATA_COMPRESSION = PAGE);
```

Kompresi tipe PAGE mengurangi ukuran penyimpanan dan mempercepat akses I/O pada tabel Fakta_Pengiriman_Kendaraan yang sangat besar volumenya.

- Hasil Transformasi ke Tabel Dimensi (Dim_Wilayah)

```
CREATE TABLE Dim_Wilayah (
    wilayah_id SERIAL PRIMARY KEY,
    city TEXT,
    region TEXT,
    country TEXT
);
```

c. Tier 3: Analysis dan Reporting Layer

Lapisan ini menyediakan data yang sudah siap untuk dianalisis, baik oleh pengguna bisnis maupun sistem visualisasi (BI tools). Data yang disediakan umumnya dalam bentuk agregasi, materialized views, atau data mart.

- Indexed View untuk Analitik

```
CREATE VIEW vw_AvgRatingPerModel
WITH SCHEMABINDING AS
SELECT
    m.model,
    AVG(f.avg_customer_rating) AS avg_rating
FROM dbo.Fakta_Pengiriman_Kendaraan f
JOIN dbo.Dim_Model m ON f.model_id = m.model_id
GROUP BY m.model;

CREATE UNIQUE CLUSTERED INDEX idx_rating_model
ON vw_AvgRatingPerModel (model);
```

View vw_AvgRatingPerModel menyediakan rata-rata rating pelanggan per model kendaraan. Indexed view ini mempercepat kueri pelaporan tanpa harus membaca tabel besar setiap saat.

- Data Mart

```
1. CREATE TABLE DataMart_KepuasanPelanggan AS
2. SELECT
3.     f.delivery_date,
4.     m.model,
5.     r.region,
6.     f.avg_customer_rating,
7.     f.service_visits
8. FROM Fakta_Pengiriman_Kendaraan f
9. JOIN Dim_Model m ON f.model_id = m.model_id
10.    JOIN Dim_Wilayah r ON f.wilayah_id =
   r.wilayah_id
11.    WHERE f.avg_customer_rating IS NOT NULL;
```

DataMart_KepuasanPelanggan adalah subset warehouse yang fokus pada analisis loyalitas dan kepuasan pelanggan. Disusun untuk mempermudah akses cepat oleh user tanpa query kompleks.

2.3 Design and implement partitioned tables and views

Untuk memastikan performa dan skalabilitas sistem data warehouse seiring dengan pertumbuhan data historis, proyek ini menerapkan partisi tabel dan view yang terindeks. Pendekatan ini memungkinkan eksekusi kueri yang lebih cepat dan manajemen data historis yang efisien, terutama untuk data skala besar seperti pengiriman kendaraan per tahun.

a. Partisi Tabel Fakta

Partisi dilakukan secara horizontal (row partitioning) pada tabel Fakta_Pengiriman_Kendaraan berdasarkan atribut delivery_year, yang berasal dari kolom delivery_date. Ini mempermudah pelacakan dan agregasi berdasarkan waktu. Tujuan Partisi ini untuk :

- Mempercepat kueri waktu-spesifik seperti analisis tahunan.
- Mempermudah manajemen historis (arsip dan pemeliharaan).
- Mendukung optimalisasi performa saat digunakan bersama dengan clustered index.

```
1. -- Membuat fungsi partisi berdasarkan tahun
2. CREATE PARTITION FUNCTION pf_DeliveryYear (INT)
3. AS RANGE LEFT FOR VALUES (2018, 2020, 2022, 2024);
4.
5. -- Skema partisi
6. CREATE PARTITION SCHEME ps_DeliveryYear
7. AS PARTITION pf_DeliveryYear ALL TO ([DW_Fakta]);
8.
9. -- Menerapkan indeks terpartisi pada tabel fakta
10.   CREATE CLUSTERED INDEX idx_delivery_date
11.   ON Fakta_Pengiriman_Kendaraan (delivery_date)
12.   ON ps_DeliveryYear (delivery_year);
```

Penjelasan :

- Data dikelompokkan ke dalam partisi berdasarkan delivery_year.

- Partisi dikombinasikan dengan clustered index untuk mempercepat pencarian berdasarkan tanggal.
- DW_Fakta digunakan sebagai target filegroup untuk partisi, sesuai dengan struktur storage.

b. Perancangan Indexed Views (Zona Analitik/Gold Layer)

Untuk mendukung agregasi yang sering digunakan dalam laporan dan analisis dashboard, sistem menggunakan indexed views yang menggabungkan fakta dan dimensi, serta sudah terindeks agar siap pakai.

- View Agregat : Total kendaraan per Model per Tahun

```
1. CREATE VIEW vw_TotalKendaraanPerTahun
2. WITH SCHEMABINDING AS
3. SELECT
4.    m.model,
5.    w.year,
6.    SUM(f.units_delivered) AS total_units
7. FROM dbo.Fakta_Pengiriman_Kendaraan f
8. JOIN dbo.Dim_Model m ON f.model_id = m.model_id
9. JOIN dbo.Dim_Waktu w ON f.waktu_id = w.waktu_id
10.   GROUP BY m.model, w.year;
11.
12.   CREATE UNIQUE CLUSTERED INDEX idx_units_per_year
13.   ON vw_TotalKendaraanPerTahun (model, year);
```

Penjelasan :

- View ini merangkum jumlah kendaraan yang dikirim per model setiap tahun.
- WITH SCHEMABINDING diperlukan untuk indexed views.
- View ini ditempatkan di Tier 3 untuk akses cepat oleh pelaporan bisnis.

3. Pemeliharaan Struktur Indeks, Partisi, dan Strategi Penyimpanan

Untuk memastikan kinerja jangka panjang dan skalabilitas sistem data warehouse, pemilihan struktur indeks, partisi, dan strategi penyimpanan harus mempertimbangkan tidak hanya efisiensi akses data, tetapi juga kemudahan pemeliharaan dan keselarasan dengan desain star schema serta arsitektur three-tier.

3.1 Struktur Indeks

a. Tujuan

Struktur indeks dirancang untuk mengoptimalkan kecepatan kueri terhadap tabel fakta dan dimensi, khususnya saat melakukan join, agregasi, dan filter.

b. Jenis Indeks

Jenis Indeks	Target Tabel	Fungsi
Clustered Index	Fakta_Pengirim_Kendaraan	Mempercepat akses berdasarkan delivery_date
Non-Clustered Index	model_id, wilayah_id,	Mengoptimasi join ke tabel

	pelanggan_id	dimensi.
Indexed View	View agregasi per model dan tahun	Mempercepat analisis agregasi rutin pada dashboard

Ketentuan :

1. Clustered index diterapkan pada kolom waktu (delivery_date) karena banyak digunakan untuk filtering dan pengurutan.
2. Non-clustered index difokuskan pada foreign key untuk mendukung performa join antar tabel star schema.
3. Indexed views mendukung tier analitik tanpa harus mengakses tabel utama setiap saat.

3.2 Strategi Partisi

a. Tujuan

Partisi memungkinkan pengelolaan data historis dan analitik waktu-spesifik dengan lebih efisien.

b. Skema Partisi

1. Menggunakan horizontal partitioning berdasarkan delivery_year.
2. Diterapkan pada tabel fakta, yang memiliki volume data besar dan bersifat time-series.
3. Menggunakan PARTITION FUNCTION dan PARTITION SCHEME untuk SQL Server.

c. Manfaat

1. Kinerja kueri meningkat untuk analisis berbasis waktu.
2. Proses backup dan arsitektur pemeliharaan menjadi modular.
3. Operasi seperti purging, archiving, dan partition switching lebih terkontrol.

3.3 Strategi Penyimpanan Data

a. Pendekatan Fisik

terdapat 2 pendekatan fisik yaitu ;

1. Pemisahan filegroup, dimana ;
   - PRIMARY menyimpan tabel dimensi (stabil, volume kecil).
   - DW_Fakta menyimpan tabel fakta (bervolume besar, bertumbuh).
2. Kompresi PAGE diterapkan untuk mengurangi ukuran data dan mempercepat proses baca-tulis.

b. Penyesuaian dengan Arsitektur

Layer Warehouse (Tier 2) menjadi fokus utama implementasi strategi ini karena merupakan tempat penyimpanan data transaksional historis yang besar, menjadikannya krusial untuk analisis waktu-spesifik. Keselarasan strategi ini diperkuat dengan penggunaan Filegroup DW_Fakta dalam PARTITION SCHEME, yang memungkinkan pengelolaan data berdasarkan skema partisi yang telah ditentukan. Dengan demikian, indeks dan partisi tidak hanya berdiri sendiri, melainkan saling mendukung untuk mengoptimalkan performa kueri yang membutuhkan data spesifik berdasarkan rentang waktu.

4. ETL Pipeline (Extract, Transform, Load)

ETL pipeline adalah proses inti dalam pengelolaan data warehouse yang menghubungkan data mentah dari berbagai sumber dengan sistem analitik dan pelaporan. Proses ini bertujuan untuk menyiapkan data dalam bentuk yang bersih, terstruktur, dan siap digunakan dalam analisis.

4.1 Arsitektur ETL yang digunakan

ETL dalam proyek ini diimplementasikan secara modular dan terintegrasi ke dalam Tier 2 (Staging, ETL, dan Warehouse) dalam arsitektur Three-Tier. Prosesnya dibagi menjadi tiga tahap utama:

a. Extract – Pengambilan data dari sumber eksternal (CSV, API, sistem operasional).

b. Transform – Proses pembersihan, standarisasi, dan pengayaan data.

c. Load – Memasukkan data ke tabel staging, kemudian ke dimensi dan fakta.

4.2 Detail Proses ETL dan SQL Implementasi berdasarkan Dataset

a. Extract: Mengambil Data dari Sumber

```
COPY Staging_Tesla_Deliveries FROM
'/data/tesla_vehicle_deliveries.csv' DELIMITER ',' CSV
HEADER;
```

Memuat data mentah dari file CSV ke tabel staging. Semua kolom dimasukkan untuk transformasi lebih lanjut.

b. Transform: Memproses dan Menyiapkan Data Dimensi

```
-- Memasukkan data unik ke Dim_Model
INSERT INTO Dim_Model (model, battery_type,
drive_type, color)
SELECT DISTINCT Model, "Battery Type", "Drive Type",
Color
FROM Staging_Tesla_Deliveries
WHERE Model IS NOT NULL;
```

Menyaring kombinasi unik atribut model kendaraan dan menyimpannya di tabel Dim_Model.

c. Memasukkan data waktu

```
-- Memasukkan data waktu ke Dim_Waktu
INSERT INTO Dim_Waktu (year, quarter)
SELECT DISTINCT Year,
    CASE WHEN Quarter = 'Q1' THEN 1
         WHEN Quarter = 'Q2' THEN 2
         WHEN Quarter = 'Q3' THEN 3
         WHEN Quarter = 'Q4' THEN 4 END
FROM Staging_Tesla_Deliveries;

```

Penjelasan : Mengonversi kuartal ke bentuk numerik dan menyimpannya bersama tahun di Dim_Waktu.

d. Memasukkan wilayah

```
-- Memasukkan wilayah ke Dim_Wilayah
INSERT INTO Dim_Wilayah (region)
SELECT DISTINCT Region FROM Staging_Tesla_Deliveries;
```

Penjelasan : Menyimpan daftar wilayah distribusi kendaraan unik ke dalam Dim_Wilayah.

e. Menyiapkan data performa teknis

```
-- Menyiapkan data performa teknis ke Dim_Performa
INSERT INTO Dim_Performa (software_version,
efficiency, charging_time, autopilot_enabled)
SELECT DISTINCT "Software Version", "Efficiency
(kWh/100km)", "Charging Time (Min)", "Autopilot
Enabled"
FROM Staging_Tesla_Deliveries;
```

Penjelasan : Mengambil data unik performa teknis dari staging dan menyimpannya sebagai entitas Dim_Performa.

f. Menyiapkan data pelanggan

```
-- Menyiapkan data pelanggan ke Dim_Pelanggan
INSERT INTO Dim_Pelanggan (warranty_period,
update_frequency, customer_rating)
SELECT DISTINCT "Warranty (Years)", "Update Frequency
(Per Year)", "Avg Customer Rating"
FROM Staging_Tesla_Deliveries;
```

Penjelasan : Menyusun informasi terkait pelanggan berdasarkan garansi, update, dan rating ke dalam Dim_Pelanggan.

4.3 Load

a. Memasukkan Data ke Tabel Fakta

```
INSERT INTO Fakta_Pengiriman_Kendaraan (
    model_id, wilayah_id, waktu_id, performa_id,
pelanggan_id,
    units_delivered, production_delay_days,
average_price,
    service_visits, battery_replacement_rate,
total_miles_driven,
    number_of_recalls, CO2_saved, resale_value,
avg_customer_rating
)
SELECT
    m.model_id,
    r.wilayah_id,
      w.waktu_id,
      p.performa_id,
      c.pelanggan_id,
```

```
    s."Units Delivered",
    s."Production Delay (Days)",
    s."Average Price (USD)",
    s."Service Visits (Per Year)",
    s."Battery Replacement Rate",
    s."Total Miles Driven",
    s."Number of Recalls",
    s."CO2 Saved (Tons)",
    s."Resale Value (%)",
    s."Avg Customer Rating"
FROM Staging_Tesla_Deliveries s
JOIN Dim_Model m ON s.Model = m.model
JOIN Dim_Wilayah r ON s.Region = r.region
JOIN Dim_Waktu w ON s.Year = w.year AND
   CASE WHEN s.Quarter = 'Q1' THEN 1
        WHEN s.Quarter = 'Q2' THEN 2
        WHEN s.Quarter = 'Q3' THEN 3
        WHEN s.Quarter = 'Q4' THEN 4 END = w.quarter
JOIN Dim_Performa p ON s."Software Version" =
  p.software_version
JOIN Dim_Pelanggan c ON s."Avg Customer Rating" =
  c.customer_rating;
```

Data staging dimuat ke dalam tabel fakta Fakta_Pengiriman_Kendaraan dengan join ke semua dimensi menggunakan nilai-nilai yang telah ditransformasi.

4.4. Tools Pendukung

ETL ini dapat dijalankan menggunakan:

- Python (pandas, psycopg2): Untuk proses extract dan transform.
- PostgreSQL atau SQL Server: Untuk load ke warehouse.
- Airflow atau cronjob: Untuk orkestrasi ETL secara terjadwal.