LAPORAN MISI KETIGA PERGUDANGAN DATA

RESTORAN DAN MAKANAN

Kelompok 17 RB :

1.	Deyvan Loxefal	121450148
2.	Allya Nurul Islami Pasha	122450033
3.	Azizah Kusumah Putri	122450068
4.	Naufal Fakhri	122450089
5.	Anwar Muslim	122450117
6.	Dhafin Razaqa Luthfi	122450133

PROGRAM STUDI SAINS DATA
FAKULTAS SAINS
INSTITUT TEKNOLOGI SUMATERA
2025

Latar Belakang

Data warehouse merupakan sistem penyimpanan data yang dirancang untuk mendukung proses analisis dan pengambilan keputusan. Data warehouse dapat digunakan untuk melakukan integrasi data dari berbagai sumber operasional menjadi satu kesatuan yang tersusun secara strategis. Pada studi kasus kali ini berfokus pada pembangunan data warehouse untuk sebuah restoran pizza. Dimana restoran pizza sendiri memiliki berbagai proses operasional yang memberikan data beragam, seperti data penjualan, produk, waktu transaksi, size pizza, dan juga bahan baku. Data warehouse berbasis skema bintang digunakan karena skema ini dapat melakukan pengorganisasian data dengan cara yang sederhana dan efektif untuk analisis data. Skema bintang terdiri dari tabel fakta untuk data transaksi dan tabel dimensi yang digunakan untuk informasi deskriptif.

Selain itu, proses ETL (Extract, Transform, Load) digunakan untuk melakukan penggabungan data dari sumber eksternal, mentransformasi data agar data dapat sesuai dengan struktur tabel, dan memasukkannya ke dalam data warehouse. Dengan ini, diharapkan proses analisis data dapat dilakukan dengan efisien dan akurat yang dapat membantu dalam melakukan pengambilan keputusan yang strategis terkait operasional restoran pizza.

1. Desain Logikal dan Implementasi Dimensi Tabel

1.1 Deskripsi Umum Tabel Dimensi

Dalam perancangan awal gudang data (data warehouse), pembangunan **tabel dimensi** merupakan langkah penting karena disinilah informasi deskriptif yang melengkapi data kuantitatif disimpan. Tabel fakta mencatat data numerik seperti jumlah transaksi, kuantitas terjual, dan total harga, sedangkan tabel dimensi menyediakan konteks seperti nama produk, kategori pizza, ukuran, bahan baku, dan waktu pembelian.

Tabel dimensi dirancang agar dapat digunakan oleh berbagai jenis pengguna, mulai dari analis data hingga manajer penjualan atau bagian logistik. Tujuan utamanya adalah mendukung kebutuhan analitik seperti:

- Mengidentifikasi pizza terlaris berdasarkan ukuran dan kategori,
- Menganalisis tren pembelian per waktu (hari, bulan, tahun),
- Menghitung kebutuhan bahan baku berdasarkan komposisi pizza.

Dengan struktur yang tertata dan hirarki data yang jelas (misalnya dari tanggal ke bulan ke tahun), pengguna dapat melihat data dari berbagai sudut pandang untuk mendukung pengambilan keputusan strategis.

1.2 Tabel Dimensi yang Dirancang

Berdasarkan struktur data pizza_sales.csv, empat tabel dimensi dibangun untuk mendukung analisis transaksi:

Nama Tabel	Atribut Utama	Deskripsi	Contoh Nilai
pizza	pizza_name_id, pizza_name, pizza_category	Data nama dan kategori pizza	pizza_001, "Margherita", "Classic"
ukuran	pizza_size, size_category	Ukuran pizza dalam label dan keterangan	"L", "Large"
waktu	order_date, tahun, bulan, hari	Waktu terjadinya transaksi	"2023-01-12", 2023, 1, 12
bahanbaku	pizza_ingredients, stok_tersedia	Bahan utama pizza dan stok (jika tersedia)	"cheese, tomato", 20

1.3 Pertimbangan Desain

1. **Konsistensi dan Kemudahan Relasi**

   Masing-masing tabel dimensi menggunakan atribut utama sebagai primary key, yang nantinya menjadi foreign key dalam tabel fakta. Dengan cara ini, setiap transaksi penjualan bisa dikaitkan dengan informasi deskriptif dari masing-masing dimensi.

2. **Struktur Hierarki untuk Waktu**

   Tabel waktu menyusun data berdasarkan hierarki:

   - order_date → bulan → tahun

   Struktur ini memungkinkan analisis agregat seperti total penjualan per bulan atau per tahun secara mudah, dan mendukung proses drill-down maupun roll-up saat melakukan OLAP.

3. **Format Data Standar**

Penggunaan format dan tipe data yang sesuai dilakukan untuk mendukung efisiensi penyimpanan serta menjaga akurasi data. Atribut order_date pada tabel waktu disimpan dengan tipe DATE menggunakan format standar YYYY-MM-DD agar dapat langsung digunakan dalam fungsi waktu dan agregasi. Untuk bahan baku, karena terdiri dari daftar beberapa komponen (seperti keju, saus, dan sayuran), maka digunakan tipe data VARCHAR yang cukup panjang agar fleksibel menampung variasi isi. Ukuran pizza yang umumnya hanya satu huruf seperti S, M, L, atau XL disimpan dalam bentuk CHAR(1) atau VARCHAR(2) agar ringkas. Sementara itu, data numerik seperti stok_tersedia pada tabel bahanbaku ditetapkan bertipe INT, mengingat nilainya bersifat kuantitatif dan dapat digunakan dalam perhitungan atau analisis logistik.

4. **Pemahaman oleh Pengguna Non-Teknis**

Pemahaman oleh pengguna non-teknis juga menjadi pertimbangan penting dalam perancangan tabel dimensi. Oleh karena itu, nama tabel dan atribut disusun dengan istilah yang sederhana, konsisten, dan mudah dimengerti, tanpa istilah teknis atau singkatan yang membingungkan. Misalnya, nama atribut seperti pizza_name atau order_date dipilih karena cukup deskriptif dan intuitif. Hal ini dimaksudkan agar struktur data tidak hanya dapat dimanfaatkan oleh analis atau pengembang, tetapi juga dapat diakses dan dipahami oleh pihak-pihak non-teknis seperti tim manajemen produk, staf pengadaan bahan baku, hingga bagian pemasaran. Dengan pendekatan ini, seluruh pemangku kepentingan dalam organisasi dapat membaca dan menggunakan data secara langsung untuk mendukung pengambilan keputusan strategis tanpa harus bergantung pada tim teknis.

1.4 Implementasi Tabel Dimensi

Desain logikal merupakan tahapan penting dalam pembangunan data warehouse, karena mengubah konsep multidimensi ke dalam bentuk tabel-tabel relasional yang siap diimplementasikan dalam sistem basis data.

```
CREATE TABLE waktu (
   order_date DATE PRIMARY KEY,
   hari VARCHAR(10),
   bulan TINYINT,
   tahun SMALLINT
);
```

```
CREATE TABLE pizza (
    pizza_name_id VARCHAR(20) PRIMARY KEY,
    pizza_name VARCHAR(100),
    pizza_category VARCHAR(50)
);

CREATE TABLE ukuran (
    pizza_size VARCHAR(10) PRIMARY KEY,
    size_category VARCHAR(50)
);
CREATE TABLE bahanbaku (
    pizza_ingredients VARCHAR(255) PRIMARY KEY,
    stok_tersedia INT
);
```

1.5 Proses Ekstraksi dan Transformasi Data ke Tabel Dimensi

Setelah perancangan tabel dimensi selesai, tahap berikutnya adalah melakukan ekstraksi data dari sumber asli pizza_sales.csv ke dalam tabel dimensi yang sudah dibuat. Proses ini melibatkan transformasi data mentah menjadi format yang sesuai dengan struktur tabel, seperti konversi tipe data, pembersihan data duplikat, dan pemetaan kolom sumber ke kolom tujuan. Contohnya, untuk tabel waktu, tanggal transaksi dipecah menjadi beberapa atribut yaitu tanggal lengkap (order_date), nama hari (hari), nomor bulan (bulan), dan tahun (tahun) guna memudahkan analisis berdasarkan waktu. Pada tabel pizza, atribut pizza_name dan pizza_category dipastikan konsistensinya dan diproses untuk menghindari entri ganda. Data bahan baku pada bahan baku juga diproses agar dapat merepresentasikan komposisi bahan secara akurat.

2. Design and Implement Fact Table

Pada tahap ini, dilakukan perancangan dan mengimplementasikan tabel fakta (fact table) yang menjadi pusat dari struktur *star schema* yang telah dikembangkan. Tabel fakta ini digunakan untuk menyimpan data transaksi penjualan yang bersifat kuantitatif serta memiliki hubungan langsung dengan beberapa tabel dimensi yang telah ditentukan sebelumnya.

2.1 Struktur tabel fakta

Tabel fakta yang dirancang dalam sistem ini dinamakan fact_penjualan, yang berfungsi sebagai pusat dalam skema *star schema* karena menyimpan data kuantitatif transaksi penjualan pizza. Tabel ini dihubungkan dengan beberapa tabel dimensi menggunakan *foreign key*.

Berikut adalah struktur kolom beserta tipe datanya:

Nama Kolom	Tipe Data	Keterangan
order_date	DATE	Foreign key yang merujuk ke waktu.order_date, menyimpan tanggal pemesanan.
order_id	INT	Primary key, ID untuk setiap transaksi penjualan.
pizza_category	VARCHAR(50)	Kategori pizza (contoh : Classic, Supreme). Disimpan langsung untuk efisiensi analisis.
pizza_ingredients	VARCHAR(255)	Foreign key yang merujuk ke bahanbaku.pizza_ingredients, berisi daftar bahan.
pizza_name_id	VARCHAR(20)	Foreign key yang merujuk ke pizza.pizza_name_id, menunjukkan jenis pizza.
pizza_size	VARCHAR(10)	Foreign key yang merujuk ke ukuran.pizza_size, menunjukkan ukuran pizza
quantity	INT	Data kuantitatif utama yang menyatakan jumlah unit pizza yang terjual.

Relasi Foreign Key dalam Tabel fact_penjualan :

```
CREATE TABLE fact_penjualan (
    order_id INT PRIMARY KEY,
    pizza_name_id VARCHAR(20),
    pizza_size VARCHAR(10),
    pizza_category VARCHAR(50),
    order_date DATE,
    pizza_ingredients VARCHAR(255),
    quantity INT,
```

```
    FOREIGN KEY (pizza_name_id) REFERENCES
    pizza(pizza_name_id),
    FOREIGN KEY (pizza_size) REFERENCES
ukuran(pizza_size),
    FOREIGN KEY (order_date) REFERENCES waktu(order_date),
    FOREIGN KEY (pizza_ingredients) REFERENCES
    bahanbaku(pizza_ingredients)
);
```

Tabel ini memiliki empat kolom foreign key yang masing-masing merujuk ke tabel dimensi: pizza, ukuran, waktu, dan bahanbaku. Kolom pizza_category ditambahkan ke dalam tabel fakta sebagai redundansi (denormalisasi) untuk mempercepat analisis tanpa perlu melakukan join tambahan ke tabel pizza.

2.2 Peran Data Kuantitatif dalam tabel fakta

Data kuantitatif merupakan elemen inti dalam tabel fakta karena menyajikan informasi numerik yang dapat digunakan dalam proses analisis, perhitungan, dan pengambilan keputusan. Dalam konteks proyek ini, data kuantitatif utama yang disimpan dalam tabel fact_penjualan adalah jumlah penjualan yang direpresentasikan oleh kolom quantity.

Kolom quantity mencerminkan jumlah unit pizza yang terjual dalam satu transaksi. Informasi ini sangat penting karena menjadi dasar untuk melakukan berbagai analisis bisnis dan operasional, antara lain untuk :

- Menghitung total penjualan per hari, minggu, atau bulan.
- Menilai performa produk berdasarkan jumlah penjualan masing-masing jenis pizza atau ukuran pizza.
- Melakukan analisis tren terhadap permintaan pelanggan dari waktu ke waktu.
- Mendukung pengambilan keputusan, misalnya dalam perencanaan stok bahan baku dan evaluasi strategi pemasaran.

Selain quantity, atribut seperti pizza_category juga dapat dikategorikan sebagai semi-kuantitatif, karena meskipun bukan angka, data ini mendukung analisis berdasarkan kelompok produk seperti "Classic", "Supreme", atau "Vegetarian". Informasi ini bisa digunakan untuk membandingkan performa antar kategori produk.

2.3 Proses ETL (Extract, Transform, Load)

Dalam implementasi ini, proses ETL digunakan untuk memindahkan data dari sumber awal (dalam bentuk CSV atau database SQLite) ke dalam tabel fact_penjualan.

- Extract: Mengambil data dari file CSV orders.csv yang berisi data transaksi mentah.
- Transform: Menyesuaikan format data agar sesuai dengan struktur tabel, seperti konversi tanggal, validasi foreign key, dan mapping nama pizza ke ID.
- Load: Memasukkan data hasil transformasi ke dalam tabel fact_penjualan menggunakan perintah INSERT.

contoh kode ETL sederhana menggunakan python :

```
import csv
import mysql.connector
from datetime import datetime

# Koneksi ke database MySQL
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="password",
    database="gudang_data"
)
cursor = db.cursor()

# Ekstraksi dan Load dari CSV
with open('orders.csv', newline='') as file:
    reader = csv.DictReader(file)
    for row in reader:
        cursor.execute("""
            INSERT INTO fact_penjualan (
                order_id, pizza_name_id, pizza_size,
                pizza_category, order_date,
                pizza_ingredients, quantity
            )
            VALUES (%s, %s, %s, %s, %s, %s, %s)
        """, (
            int(row['order_id']),
            row['pizza_name_id'],
            row['pizza_size'],
            row['pizza_category'],

            datetime.strptime(row['order_date'],
            "%Y-%m-%d").date(),
            row['pizza_ingredients'],
            int(row['quantity'])
        ))
```

```
db.commit()
cursor.close()
db.close()
```

3. Desain dan Implementasi Index

Pada bagian ini dilakukan desain dan implementasi indeks dalam gudang data pizza_sales untuk meningkatkan kinerja kueri. Tujuan utama dari desain dan implementasi index ini untuk mengeksplorasi berbagai jenis indeks, menentukan kolom prioritas untuk pengindeksan, menulis kode SQL untuk membuat indeks, dan menganalisis dampak indeks terhadap kinerja kueri.

3.1 Jenis-jenis Indeks: Clustered dan Non-Clustered

Indeks adalah struktur data yang digunakan untuk mempercepat pengambilan data dari tabel database. Mereka bekerja mirip dengan indeks buku, yang memungkinkan pengguna menemukan informasi dengan cepat tanpa harus membaca seluruh buku. Dalam database, indeks memungkinkan DBMS untuk menentukan baris tertentu dalam tabel dengan cepat tanpa memindai seluruh tabel. Ada dua jenis indeks utama :

- Clustered Index (Indeks Klaster) :

  Indeks Klaster gunanya untuk menentukan urutan fisik penyimpanan data dalam tabel. Dalam indeks klaster hanya ada satu indeks klaster per tabel. Indeks klaster sangat efisien untuk kueri yang mengambil rentang data. Berikut adalah kode SQL untuk membuat clustered index pada tabel dimensi waktu

```
CREATE CLUSTERED INDEX idx_order_date
ON waktu (order_date);
```

- Non-Clustered Index (Indeks Non-Klaster) :

  Indeks non-klaster berfungsi untuk membuat struktur yang berisi salinan kolom yang diindeks dan pointer ke lokasi data aktual. Sebuah tabel dapat memiliki banyak indeks non-klaster. Indeks klaster ini berguna untuk kueri yang mencari baris tertentu berdasarkan nilai kolom yang diindeks. Berikut adalah kode SQL untuk membuat indeks non-klaster pada tabel fakta fact_penjualan

```
CREATE INDEX idx_pizza_name_id
ON fact_penjualan (pizza_name_id);

CREATE INDEX idx_pizza_size
ON fact_penjualan (pizza_size);
```

```
CREATE INDEX idx_pizza_category
ON fact_penjualan (pizza_category);
```

3.2 Strategi Pemilihan Kolom Indeks

Pemilihan kolom untuk di indeks sangat penting untuk memaksimalkan kinerja kueri. Berikut ada beberapa strategi dan pertimbangan dalam pemilihan kolom indeks :

- Kolom yang sering digunakan dalam klausa WHERE
- Kolom yang digunakan dalam JOIN
- Kolom dengan kardinalitas tinggi

3.3 Dampak Indeks Terhadap Performa Kueri

Indeks secara signifikan meningkatkan kecepatan kueri dengan memungkinkan DBMS untuk mengambil data tertentu dengan cepat. Namun, indeks juga memiliki beberapa kelebihan seperti dapat meningkatkan ukuran database maka akan membutuhkan ruang penyimpanan tambahan. Selain itu, operasi seperti INSERT, UPDATE, dan DELETE menjadi lebih lambat karena DBMS juga harus memperbarui indeks.

3.4 Pengujian Performa Kueri

Perintah EXPLAIN dapat memberikan informasi tentang bagaimana DBMS menjalankan kueri. Ini memungkinkan kita untuk melihat apakah indeks digunakan, jenis pemindaian yang dilakukan, dan perkiraan biaya kueri. Berikut ini adalah kode SQL yang menggunakan EXPLAIN

```
EXPLAIN SELECT
    fp.order_id,
    p.pizza_name,
    u.size_category,
    w.order_date,
    fp.quantity
FROM
    fact_penjualan fp
JOIN
    pizza p ON fp.pizza_name_id = p.pizza_name_id
JOIN
    ukuran u ON fp.pizza_size = u.pizza_size
JOIN
    waktu w ON fp.order_date = w.order_date
```

```
WHERE
   fp.order_date BETWEEN '2023-01-01' AND '2023-01-31'
   AND p.pizza_name_id = 'pizza_005';
```

3.5 Pemeliharaan Indeks

Indeks dapat menjadi terfragmentasi dengan seiring waktu karena operasi INSERT, UPDATE, dan DELETE. Fragmentasi ini dapat menurunkan kinerja kueri. Oleh karena itu, penting untuk memelihara indeks secara teratur dengan mengatur ulang indeks untuk mengoptimalkan urutan data dan mengurangi fragmentasi. Selain itu dapat dilakukan Rebuild Index atau membuat ulang indeks dari awal. Berikut adalah kode SQL untuk reorganisasi dan rebuild index.

```
-- Reorganize index
ALTER TABLE fact_penjualan REORGANIZE INDEX idx_pizza_name_id;

-- Rebuild index
ALTER TABLE fact_penjualan REBUILD INDEX idx_pizza_name_id;
```

4. Desain Penyimpanan (Storage Design)

Perancangan desain penyimpanan (design storage) dalam data warehouse merupakan aspek penting yang berfokus pada bagaimana data secara fisik disimpan dan diorganisir. Tujuan utama dari desain penyimpanan ini adalah untuk mencapai efisiensi ruang penyimpanan, meningkatkan performa kueri, dan memastikan integritas data dalam sistem data warehouse.

Untuk mendukung efisiensi sistem data warehouse pada database pizza sales, strategi penyimpanan dirancang dengan memisahkan antara tabel besar dan kecil berdasarkan ukuran dan frekuensi aksesnya. Tabel fakta yang berisi data transaksi dalam jumlah besar dialokasikan ke filegroup terpisah bernama Fact_DW, sementara tabel-tabel dimensi seperti pizza, ukuran, waktu, dan bahanbaku tetap disimpan di filegroup default karena ukurannya relatif kecil dan sering digunakan dalam operasi join sehingga memerlukan akses yang cepat dan konsisten.

SQL untuk alokasi filegroup:

```
ALTER DATABASE PizzaDW
ADD FILEGROUP Fact_DW;

ALTER DATABASE PizzaDW
ADD FILE (
    NAME = 'PizzaDW_Fakta_Data',
    FILENAME = 'D:\SQLData\PizzaDW_Fakta.ndf',
    SIZE = 200MB,
    MAXSIZE = 1GB,
    FILEGROWTH = 50MB
) TO FILEGROUP Fact_DW;
```

Dengan konfigurasi ini, kita memastikan bahwa data besar dari tabel fakta tidak bercampur dengan data kecil dari dimensi, serta mendapatkan manfaat dari manajemen ruang dan performa yang lebih baik.

Kompresi data tipe PAGE digunakan untuk tabel fakta karena memberikan efisiensi ruang penyimpanan yang tinggi tanpa mengorbankan performa read.

```
ALTER TABLE sales_fakta
REBUILD WITH (DATA_COMPRESSION = PAGE);
```

Kompresi PAGE pada tabel sales_fakta dilakukan sebagai bagian dari optimasi penyimpanan di lapisan Silver dalam arsitektur Medallion. Pada tahap ini, data telah dibersihkan dan dinormalisasi ke dalam struktur tabel fakta dan dimensi, sehingga volume datanya menjadi cukup besar dan konsisten untuk dianalisis. Kompresi dapat menghemat ruang penyimpanan dan meningkatkan efisiensi I/O saat melakukan pemrosesan atau kueri analitik. Dengan demikian, strategi kompresi ini tidak hanya mengoptimalkan performa di lapisan Silver, tetapi juga mendukung kelancaran aliran data menuju lapisan Gold, di mana data sudah siap digunakan untuk agregasi, visualisasi, dan pengambilan keputusan bisnis.

Struktur Medallion diadopsi untuk memastikan kualitas dan keteraturan data dari tahap mentah hingga siap dianalisis. Untuk penjelasan selengkapnya telah dirangkum dalam tabel berikut:

Lapisan	Deskripsi
Bronze	Menyimpan data mentah dari sumber data (pizza_sales.csv)

Silver	Data telah dibersihkan dan dinormalisasi ke dalam tabel dimensi dan fakta
Gold	Data siap analisis, seperti agregat bulanan, penjualan per kategori, dan KPI lainnya

Dengan penggunaan arsitektur Medallion, proses pengelolaan data menjadi lebih terstruktur, bertahap, dan mudah dipelihara. Pendekatan ini dapat memudahkan pengendalian versi data, audit, dan pelacakan perubahan. Selain itu, strategi ini memungkinkan optimalisasi performa yang secara keseluruhan mendukung kebutuhan analitik bisnis secara efisien dan scalable.

5. Design dan Implementasi Poisis Tabel dan View

Dalam implementasi gudang data pada sistem penjualan produk pizza, strategi partisi dan view agregat menjadi penting untuk mengelola pertumbuhan volume data transaksi yang terus meningkat dari waktu ke waktu. Dengan penerapan partisi dan pandangan (view) yang tepat, sistem tidak hanya mampu menangani data historis dengan lebih efisien, tetapi juga dapat mempercepat eksekusi kueri-kueri analitik yang umum dilakukan oleh manajemen, seperti analisis penjualan bulanan, tren kategori produk, atau proyeksi kebutuhan bahan baku.

5.1 Pemilihan Struktur Partisi

A.Partisi Horizontal Berdasarkan Tanggal Pemesanan

Partisi horizontal diterapkan pada tabel penjualan sebagai tabel fakta utama dalam skema ini. Strategi partisi didasarkan pada atribut order_date, dengan pembagian per tahun. Kolom ini dipilih karena sebagian besar analisis penjualan bersifat waktu-spesifik, misalnya tren penjualan tahunan atau bulanan.

Tujuan dari strategi ini antara lain:

- Mempercepat kueri yang memfilter data berdasarkan periode tertentu (per bulan atau per tahun).
- Memisahkan data historis agar pemeliharaan, backup, dan arsip data menjadi lebih mudah.
- Menghindari pemrosesan keseluruhan tabel untuk kueri waktu yang hanya memerlukan sebagian data.

```
CREATE TABLE penjualan_partitioned (
    order_id INT,
```

```
    pizza_name_id VARCHAR(20),
    pizza_size VARCHAR(10),
    order_date DATE,
    quantity INT,
    total_price INT
)
PARTITION BY RANGE (YEAR(order_date)) (
    PARTITION p2022 VALUES LESS THAN (2023),
    PARTITION p2023 VALUES LESS THAN (2024),
    PARTITION p2024 VALUES LESS THAN (2025)
);
```

B. Strategi Pemeliharaan Partisi

Untuk menjaga efisiensi dan keteraturan data, dilakukan beberapa tindakan berkala seperti:

- Split Partition: dilakukan saat memasuki tahun baru, untuk menyiapkan slot data tahun berjalan.
- Merge Partition: digunakan jika data lama ingin digabung untuk menghemat ruang atau mempercepat pencarian agregat.
- Drop Partition: dimungkinkan bila ada data lama yang sudah diarsipkan di sistem terpisah.

5.2 Perancangan View Agregat untuk Analisis

Agar proses analitik lebih cepat dan hasil ringkasan bisa langsung dikonsumsi oleh pengguna non-teknis, dibangun beberapa view agregat yang melakukan rekapitulasi data transaksi berdasarkan dimensi waktu, kategori, dan ukuran pizza.

View Agregat Penjualan per Kategori per Tahun:

```
CREATE VIEW vw_penjualan_kategori_tahunan AS
SELECT
    p.pizza_category,
    YEAR(f.order_date) AS tahun,
    SUM(f.quantity) AS total_terjual
FROM penjualan f
JOIN pizza p ON f.pizza_name_id = p.pizza_name_id
GROUP BY p.pizza_category, YEAR(f.order_date);
```

View ini dapat digunakan oleh bagian pemasaran untuk melihat tren penjualan masing-masing kategori produk dari tahun ke tahun, tanpa harus mengakses tabel transaksi mentah.

5.3 Strategi Penyimpanan Data

Dalam arsitektur penyimpanan data warehouse berbasis produk, pendekatan multi-zona juga digunakan untuk memisahkan data mentah dari data analitik siap pakai:

- Zona Bronze (Staging): berisi data mentah dari file pizza_sales.csv, tanpa partisi, dan hanya digunakan untuk transformasi awal.
- Zona Silver (Cleaned): tabel fakta dan dimensi yang telah dibersihkan dan distandarisasi, termasuk penjualan_partitioned.
- Zona Gold (Analytic): kumpulan view agregat seperti vw penjualan kategori tahunan, yang telah dioptimasi untuk digunakan langsung oleh pengguna bisnis.

5.4 Optimasi dan Monitoring

- Kueri terhadap tahun tertentu dapat langsung ditargetkan ke partisi terkait tanpa membaca seluruh tabel transaksi, sehingga mempercepat respon sistem.
- View yang dibuat dapat ditambahkan indeks (jika engine mendukung indexed view) untuk mempercepat pemanggilan data ringkasan.
- Monitoring performa dilakukan dengan mengevaluasi waktu eksekusi kueri dan ukuran partisi tiap tahun, serta statistik pemakaian view.

5.5 Studi Kasus Simulasi Query: Analisis Musim Promosi

Salah satu kasus analitik yang sering muncul dalam bisnis makanan cepat saji adalah menganalisis dampak promosi terhadap penjualan. Dalam sistem ini, view agregat juga dirancang untuk menganalisis performa penjualan selama periode promosi tertentu, misalnya Ramadan, Tahun Baru, atau program diskon bulanan.

```
CREATE VIEW vw_penjualan_promosi AS
SELECT
   MONTH(order_date) AS bulan,
   pizza_category,
   SUM(quantity) AS total_item,
   SUM(total_price) AS omzet
FROM penjualan
WHERE MONTH(order_date) IN (3, 4)
GROUP BY pizza_category, MONTH(order_date);
```

View ini memudahkan tim pemasaran mengevaluasi apakah program promosi benar-benar meningkatkan volume penjualan di tiap kategori produk.

5.6 Keterkaitan dengan Sistem Operasional

Meski partisi dan view dirancang dalam gudang data, hasilnya tidak berdiri sendiri. View agregat seperti vw_penjualan_kategori_tahunan dapat dihubungkan kembali ke sistem operasional, misalnya dengan:

- Menggunakan view sebagai dasar penjadwalan pembelian bahan baku.
- Menyediakan informasi real-time untuk dashboard kasir cabang atau POS (Point of Sale). Menghasilkan laporan mingguan otomatis yang dikirim ke email supervisor cabang.