**LAPORAN RESMI**

**MODUL II**

**VIEW**



**NAMA : ANISYAFAAH**

**N.R.P : 220441100105**

**DOSEN : FITRI DAMAYANTI, S.Kom., M.Kom.**

**ASISTEN : AFFAN MAULANA ZULKARNAIN**

**TGL PRAKTIKUM : 29 MARET 2024**

**Disetujui : 23 April 2024**

**Asisten**

**AFFAN MAULANA ZULKARNAIN**

**20.04.411.00052**

**LABORATORIUM BISNIS INTELIJEN SISTEM**

**PRODI SISTEM INFORMASI**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

# **BAB I PENDAHULUAN**

## **Latar Belakang**

Dalam era digital yang semakin berkembang, penggunaan database telah menjadi suatu keharusan bagi berbagai organisasi dan perusahaan. Pertumbuhan data yang cepat dan kompleksitas informasi yang semakin tinggi membuat pentingnya pengelolaan data secara efisien. Pertama, database memungkinkan penyimpanan data dalam format terstruktur, yang memudahkan pengelolaan dan pengambilan informasi. Dengan menggunakan database, informasi dapat disimpan secara teratur dalam tabel dan kolom, memfasilitasi proses pencarian dan analisis data dengan lebih mudah dan cepat.

Selain itu, implementasi database juga memberikan keuntungan dalam hal konsistensi dan integritas data. Dengan aturan yang ditetapkan dalam basis data, seperti constraint dan relasi antartabel, kesalahan dalam input data dapat diminimalkan. Hal ini penting untuk menjaga keakuratan informasi dan mencegah terjadinya inkonsistensi yang dapat merugikan dalam pengambilan keputusan. Dengan demikian, penggunaan database membantu organisasi untuk menjaga kualitas dan keandalan data mereka.

Dalam menggunakan sebuah database, kita pasti perlu menggunakan view. Penggunaan view dalam database memberikan fleksibilitas dalam pengelolaan dan presentasi data. View memungkinkan pengguna untuk membuat tampilan virtual dari satu atau beberapa tabel, yang dapat disesuaikan dengan kebutuhan pengguna tertentu. Dengan menggunakan view, pengguna dapat mengakses dan menganalisis data tanpa harus mengetahui struktur tabel yang kompleks di baliknya.

## **Tujuan**

* Mampu memahami konsep dasar view di dalam basis data
* Mampu memahami penerapan view
* Mampu menyelesaikan pengambilan data dengan menggunakan pendekatan view

# **BAB II DASAR TEORI**

## **View**

View adalah tabel virtual yang terbuat dari suatu query terhadap satu tabel atau beberapa tabel. View tidak ada secara nyata dalam database. View hanya digunakan untuk menyederhanakan dan mempermudah persepsi pengguna dalam database. Tidak seperti pada umumnya tabel di dalam basis data relasional, view bukanlah bagian dari skema fisik. View bersifat dinamis, ia mengandung data dari tabel yang direpresentasikannya. Dengan demikian, ketika tabel yang menjadi sumber datanya berubah, data di view juga akan berubah.

Kegunaan view antara lain:

1. Memfokuskan pada data tertentu
2. Penyederhanaan manipulasi data
3. Menyesuaikan data dengan kebutuhan user
4. Export dan impor data
5. Mengkombinasikan data terpartisi







## **Updatable View**

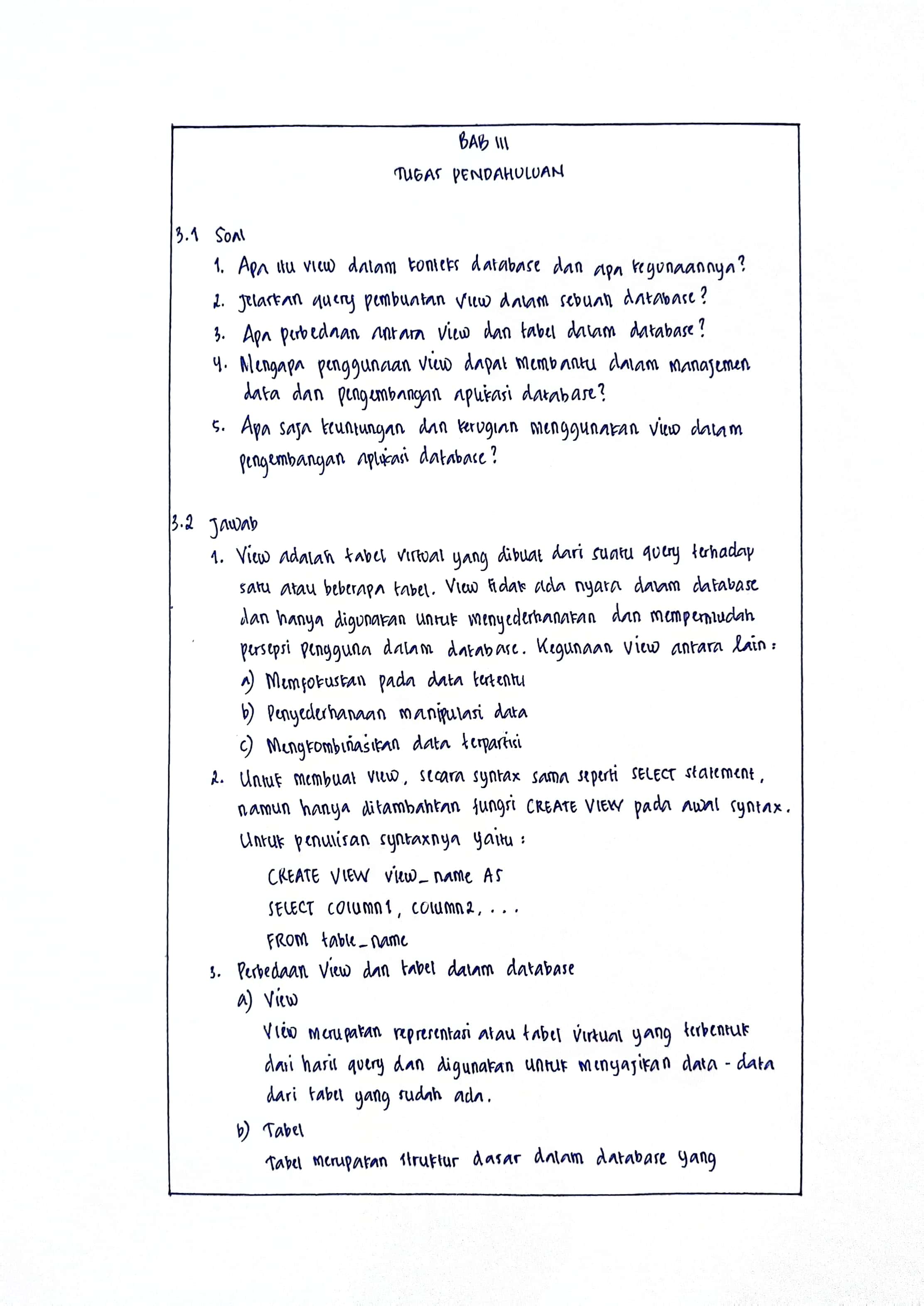
View dapat berisi read-only atau updatable. Kondisi ini sangat dipengaruhi oleh adanya pendefinisian view itu sendiri. Bagaimanapun, untuk menciptakan updatable view, pernyataan SELECT yang didefinisikan di view harus mengikuti aturan-aturan sebagai berikut:

* Pernyataan SELECT tidak boleh merujuk ke lebih dari satu tabel.
* Pernyataan SELECT tidak boleh menggunakan klausa GROUP BY atau HAVING.
* Pernyataan SELECT harus tidak menggunakan DISTINCT.
* Pernyataan SELECT harus tidak merujuk ke view lain yang tidak updatable.
* Pernyataan SELECT tidak boleh mengandung ekspresi apa pun, misalnya fungsi agregat.

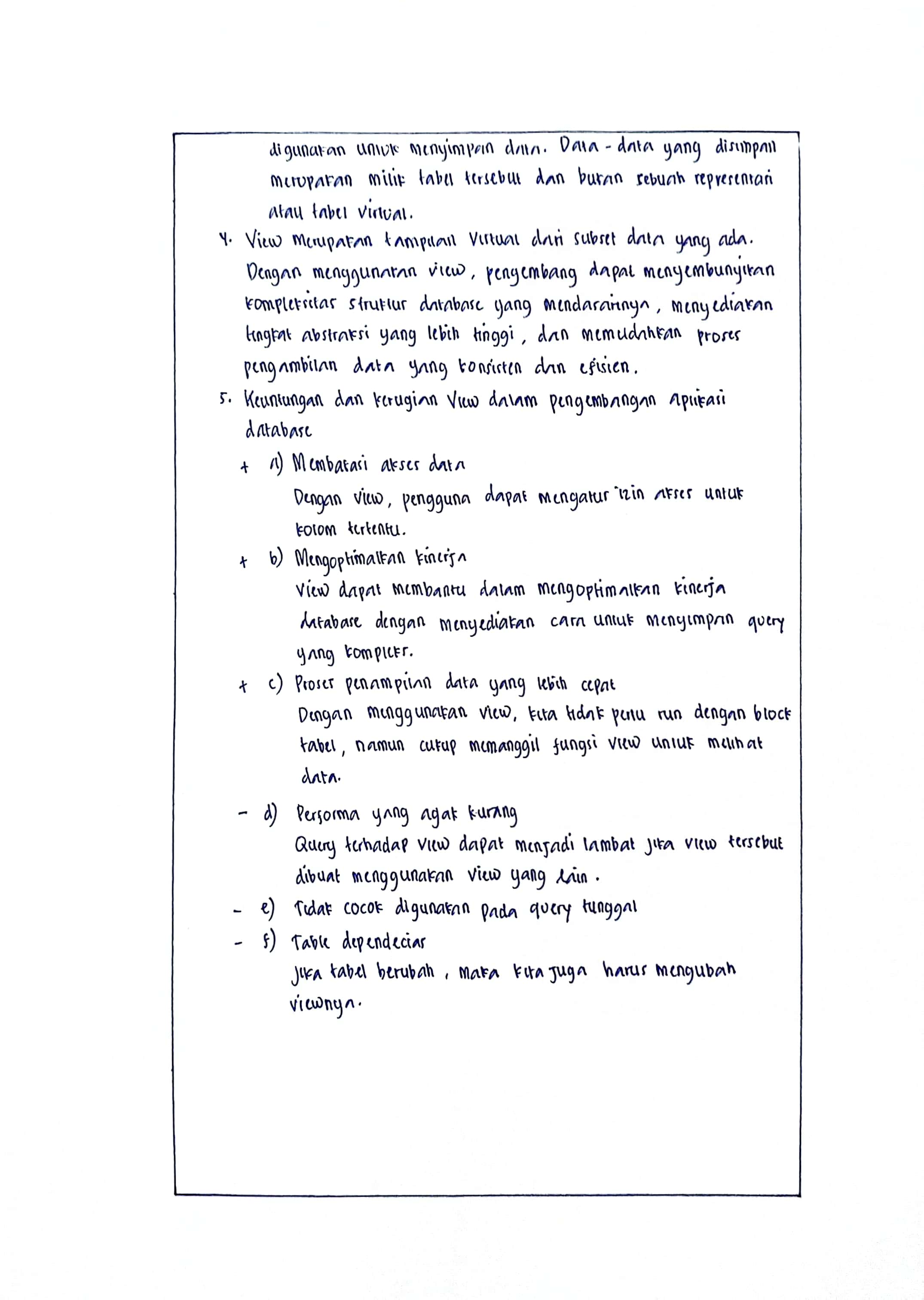
Pada hakikatnya, jika sistem database mampu menentukan pemetaan balik dari skema view ke skema tabel dasar, maka view memungkinkan untuk di update. Dalam kondisi ini, operasi-operasi INSERT, UPDATE dan DELETE dapat diterapkan pada view.

## **Hapus View**

Suatu view dari query selalu menampilkan data yang terbaru, sebagai contoh bila kita memodifikasi sebagian tupel dalam tabel dasarnya dimana view tersebut didefinisikan maka secara otomatis akan berpengaruh pada view di query. Jika tidak membutuhkan view lagi, kita bisa menggunakan perintah :

****



****

# **BAB IV IMPLEMENTASI**

## **Source Code**

### **Mengisi Data Setiap Tabel**

* **Tabel Pelanggan**

1. **Source Code**

|  |
| --- |
| CREATE DATABASE db\_penjualan;  USE db\_penjualan;  CREATE TABLE Pelanggan (  ID\_Pelanggan INT (11) PRIMARY KEY,  Nama\_Pelanggan VARCHAR (100),  Email VARCHAR (50),  Alamat VARCHAR (255)  );    INSERT INTO Pelanggan VALUES (01, "Anisyafaah", "anisyafaah@gmail.com", "Bangkalan");  INSERT INTO Pelanggan VALUES (02, "Herdiyanti Fifin Purwaningrum", "herdiyantififin@gmail.com", "Gresik");  INSERT INTO Pelanggan VALUES (03, "Anisah Nuril Izzati", "anisahnuril@gmail.com", "Bangkalan");  INSERT INTO Pelanggan VALUES (04, "Adhelia Kusumawati", "adheliakusumawati@gmail.com", "Gresik");  INSERT INTO Pelanggan VALUES (05, "Firdausi Putri Cahyani", "firdausiputri@gmail.com", "Surabaya");  INSERT INTO Pelanggan VALUES (06, "Rayhanza Nadhif Athala", "rayhanzanadhif@gmail.com", "Bangkalan");  INSERT INTO Pelanggan VALUES (07, "Birrur Rijaal", "birrurrijaal@gmail.com", "Gresik");  INSERT INTO Pelanggan VALUES (08, "Juanzha Nanda Pratama", "juanzhananda@gmail.com", "Gresik");  INSERT INTO Pelanggan VALUES (09, "Erick Firmansyah", "erickfirmansyah@gmail.com", "Surabaya");  INSERT INTO Pelanggan VALUES (010, "Syahrul Ramadhani", "syahrulramadhani@gmail.com", "Surabaya"); |

1. **Penjelasan**

Kode di atas digunakan untuk membuat tabel pelanggan dan mengisi data sebanyak 10 data. Untuk membuat tabel pelanggan menggunakan perintah CREATE TABLE Pelanggan (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id\_pelanggan sebagai primary key, nama pelanggan, email pelanggan, dan alamat pelanggan. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

* **Tabel Pesanan**

1. **Source Code**

|  |
| --- |
| CREATE TABLE Pesanan (  ID\_Pesanan INT (11) PRIMARY KEY,  ID\_Pelanggan INT (11),  Tanggal\_Pesanan DATE,  Total INT (11),  FOREIGN KEY (ID\_Pelanggan) REFERENCES Pelanggan (ID\_Pelanggan)  );    INSERT INTO Pesanan VALUES (1001, 01, "2024-04-04", 15000);  INSERT INTO Pesanan VALUES (1002, 02, "2024-04-03", 15000);  INSERT INTO Pesanan VALUES (1003, 03, "2024-04-02", 18000);  INSERT INTO Pesanan VALUES (1004, 04, "2024-04-01", 4000);  INSERT INTO Pesanan VALUES (1005, 05, "2024-03-31", 42000);  INSERT INTO Pesanan VALUES (1006, 06, "2024-03-30", 8000);  INSERT INTO Pesanan VALUES (1007, 07, "2024-03-29", 8000);  INSERT INTO Pesanan VALUES (1008, 08, "2024-03-28", 24000);  INSERT INTO Pesanan VALUES (1009, 09, "2024-03-27", 25000);  INSERT INTO Pesanan VALUES (1010, 010, "2024-03-26", 30000); |

1. **Penjelasan**

Kode di atas digunakan untuk membuat tabel pesanan dan mengisi data sebanyak 10 data. Untuk membuat tabel pesanan menggunakan perintah CREATE TABLE Pesanan (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id\_pesanan sebagai primary key, id\_pelanggan yang terhubung dengan tabel pelanggan menggunakan perintah FOREIGN KEY, tanggal pesanan, dan total harga pesanan. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO**.**

* **Tabel Produk**

1. **Source Code**

|  |
| --- |
| CREATE TABLE Produk (  ID\_Produk INT (11) PRIMARY KEY,  Nama\_Produk VARCHAR (100),  Harga INT (11),  Stok INT (11)  );  INSERT INTO Produk VALUES (111, "Susu Ultramilk", 5000, 4);  INSERT INTO Produk VALUES (112, "Richeese Nabati", 3000, 10);  INSERT INTO Produk VALUES (113, "Wafer Tango", 6000, 8);  INSERT INTO Produk VALUES (114, "Nabati Rolls", 2000, 15);  INSERT INTO Produk VALUES (115, "Susu Indomilk", 7000, 3);  INSERT INTO Produk VALUES (116, "Sosis So Nice", 4000, 6);  INSERT INTO Produk VALUES (117, "Richoco", 2000, 11);  INSERT INTO Produk VALUES (118, "Roma Malkist", 8000, 7);  INSERT INTO Produk VALUES (119, "Qtela Singkong", 8000, 10);  INSERT INTO Produk VALUES (120, "Sari Gandum", 5000, 13); |

1. **Penjelasan**

Kode di atas digunakan untuk membuat tabel produk dan mengisi data sebanyak 10 data. Untuk membuat tabel produk menggunakan perintah CREATE TABLE Produk (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id\_produk sebagai primary key, nama produk, harga produk, dan stok produk. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO**.**

* **Tabel Detail Pesanan**

1. **Source Code**

|  |
| --- |
| CREATE TABLE Detail\_Pesanan (  ID\_Detail INT (11) PRIMARY KEY,  ID\_Pesanan INT (11),  ID\_Produk INT (11),  Jumlah INT (11),  FOREIGN KEY (ID\_Pesanan) REFERENCES Pesanan (ID\_Pesanan),  FOREIGN KEY (ID\_Produk) REFERENCES Produk (ID\_Produk)  );    INSERT INTO Detail\_Pesanan VALUES (121, 1001, 111, 3);  INSERT INTO Detail\_Pesanan VALUES (122, 1002, 112, 5);  INSERT INTO Detail\_Pesanan VALUES (123, 1003, 113, 3);  INSERT INTO Detail\_Pesanan VALUES (124, 1004, 114, 2);  INSERT INTO Detail\_Pesanan VALUES (125, 1005, 115, 2);  INSERT INTO Detail\_Pesanan VALUES (126, 1006, 116, 2);  INSERT INTO Detail\_Pesanan VALUES (127, 1007, 117, 4);  INSERT INTO Detail\_Pesanan VALUES (128, 1008, 118, 3);  INSERT INTO Detail\_Pesanan VALUES (129, 1009, 119, 5);  INSERT INTO Detail\_Pesanan VALUES (130, 1010, 120, 6); |

1. **Penjelasan**

Kode di atas digunakan untuk membuat tabel detail pesanan dan mengisi data sebanyak 10 data. Untuk membuat tabel pesanan menggunakan perintah CREATE TABLE Detail\_Pesanan (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id\_detail sebagai primary key, id\_pesanan yang terhubung dengan tabel pesanan menggunakan perintah FOREIGN KEY, id\_produk yang terhubung dengan tabel produk menggunakan perintah FOREIGN KEY, dan jumlah pesanan. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO**.**

### **View Data Pesanan yang Lebih dari Rata-Rata**

1. **Source Code**

|  |
| --- |
| -- Nomor 1  SELECT AVG(Total) FROM Pesanan;  CREATE VIEW Pemesanan AS  SELECT Pelanggan.Nama\_Pelanggan, Pesanan.Total AS  Total\_Harga\_Pesanan, Pesanan.Tanggal\_Pesanan  FROM Pelanggan JOIN Pesanan ON Pelanggan.ID\_Pelanggan = Pesanan.ID\_Pelanggan  WHERE Pesanan.Total > (SELECT AVG(Total) FROM Pesanan);  SELECT \* FROM Pemesanan; |

1. **Penjelasan**

Kode di atas digunakan untuk membuat view. Karena yang ingin ditampilkan adalah pesanan yang lebih dari rata-rata, maka membutuhkan perintah AVG (Average) dari kolom total pada tabel pesanan. Selanjutnya membuat view dengan perintah CREATE VIEW nama\_view AS statement SELECTnya. View ini membutuhkan dua tabel yaitu pelanggan (nama pelanggan) dan pesanan (total dan tanggal pesanan), sehingga perlu menggunakan JOIN ON pada kedua tabel yang saling berhubungan dimana WHERE (kondisinya) berupa perintah AVG yang sebelumnya.

### **View Data Produk yang Terjual Setiap Pesanan**

1. **Source Code**

|  |
| --- |
| -- Nomor 2  CREATE VIEW Total\_Pendapatan\_Produk AS  SELECT Produk.Nama\_Produk, Produk.Harga AS Harga\_Satuan, Detail\_Pesanan.Jumlah AS Jumlah\_Produk\_Terjual, (Produk.Harga \* Detail\_Pesanan.Jumlah) AS Total\_Pendapatan  FROM Produk JOIN Detail\_Pesanan ON Produk.ID\_Produk = Detail\_Pesanan.ID\_Produk;  SELECT \* FROM Total\_Pendapatan\_Produk; |

1. **Penjelasan**

Kode view di atas digunakan untuk menampilkan total pendapatan pada keseluruhan produk. Kode ini dimulai dengan perintah membuat view yaitu CREATE VIEW nama\_view AS statement SELECTnya. View ini membutuhkan dua tabel yaitu produk (nama produk dan harga produk) dan tabel detail pesanan (jumlah), sehingga perlu menggunakan JOIN ON pada kedua tabel yang saling berhubungan. Karena ingin menampilkan total pendapatan setiap produk, maka perlu rumus total harga pada tabel pesanan \* jumlah pesanan pada tabel detail pesanan. Rumus ini dianggap sebagai total pendapatannya (AS total\_pendapatan).

### **View Data Stok yang Kurang dari 5**

1. **Source Code**

|  |
| --- |
| -- Nomor 3  CREATE VIEW Sisa\_Stok AS  SELECT Produk.Nama\_Produk, Produk.Stok AS Jumlah\_Stok FROM Produk WHERE Stok < 5;  SELECT \* FROM Sisa\_Stok; |

1. **Penjelasan**

Kode view di atas digunakan untuk menampilkan produk-produk dengan stok yang kurang dari 5. Untuk membuatnya maka memerlukan perintah CREATE VIEW nama\_view AS statement SELECTnya. View ini membutuhkan satu tabel saja yaitu tabel produk dari kolom nama produk dan stok dimana WHERE (kondisinya) diatur < 5 (kurang dari 5).

### **View Data Total Pesanan Sebulan Terakhir**

1. **Source Code**

|  |
| --- |
| -- Nomor 4  CREATE VIEW Jumlah\_Pesanan\_Sebulan AS  SELECT Pelanggan.Nama\_Pelanggan, COUNT(Pesanan.ID\_Pesanan) AS Total\_pesanan  FROM Pelanggan  JOIN Pesanan ON Pelanggan.ID\_Pelanggan = Pesanan.ID\_Pelanggan  WHERE Pesanan.Tanggal\_Pesanan BETWEEN DATE\_SUB(CURRENT\_DATE(), INTERVAL 1 MONTH) AND CURRENT\_DATE()  GROUP BY Pelanggan.ID\_Pelanggan;  SELECT \* FROM Jumlah\_Pesanan\_Sebulan; |

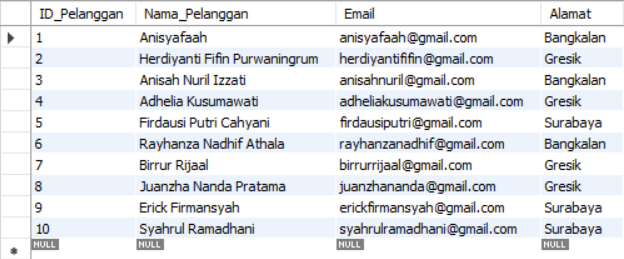
1. **Penjelasan**

Kode view di atas digunakan untuk menampilkan jumlah pesanan setiap pelanggan selama periode satu bulan terakhir. Kode ini dimulai dengan perintah membuat view yaitu CREATE VIEW nama\_view AS statement SELECTnya. View ini membutuhkan perintah COUNT (jumlah) id pesanan pada tabel pesanan sebagai total pesanannya. View ini juga membutuhkan dua tabel yaitu pelanggan (nama pelanggan), dan tabel pesanan (jumlah id pesanan), sehingga perlu menggunakan JOIN ON pada kedua tabel yang saling berhubungan. Karena ingin menampilkan total pesanan setiap pelanggan selama satu bulan terakhir, maka perlu sebuah kondisi (WHERE) Pesanan.Tanggal\_Pesanan BETWEEN DATE\_SUB(CURRENT\_DATE(), INTERVAL 1 MONTH) AND CURRENT\_DATE(). Perintah ini akan mengambil pesanan yang memiliki tanggal pemesanan antara satu bulan sebelum tanggal saat ini dan tanggal saat ini. Kemudian kode GROUP BY Pelanggan.ID\_Pelanggan untuk mengelompokkan hasil berdasarkan ID\_Pelanggan untuk menghitung jumlah pesanan yang dilakukan oleh setiap pelanggan secara terpisah.

## **Hasil**

### **Data Setiap Tabel**

* **Tabel Pelanggan**



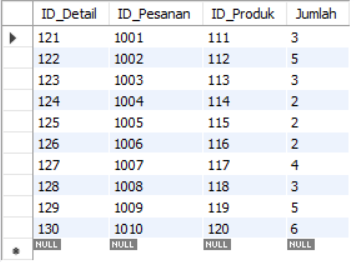
* **Tabel Pesanan**



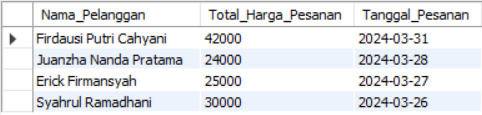
* **Tabel Produk**



* **Tabel Detail Pesanan**



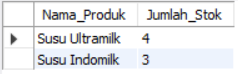
### **Data Pesanan yang Lebih dari Rata-Rata**



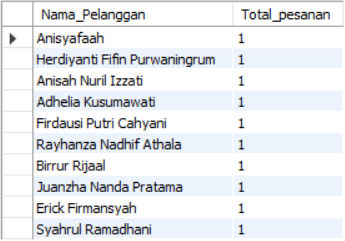
### **Data Produk yang Terjual Setiap Pesanan**



### **Data Stok yang Kurang dari 5**



### **Data Total Pesanan Sebulan Terakhir**



# **BAB V PENUTUP**

## **Analisa**

Dari hasil praktikum, praktikan menganalisa bahwa penggunaan view dalam sebuah database memiliki beragam manfaat yang signifikan. Pertama, view memungkinkan pengguna untuk mengakses dan menganalisis data dengan lebih efisien. Dengan menggunakan view, pengguna dapat membuat tampilan virtual dari data yang relevan, tanpa harus mengetahui struktur tabel yang kompleks di baliknya. Hal ini memudahkan pengguna dalam melakukan query dan analisis data, karena mereka dapat fokus pada informasi yang benar-benar mereka perlukan. Selain itu, view juga dapat membantu dalam melindungi keamanan data dengan membatasi akses pengguna hanya pada informasi yang relevan.

Kedua, penggunaan view juga meningkatkan fleksibilitas dalam pengelolaan dan presentasi data. Sebuah view dapat dibuat dari satu atau beberapa tabel, serta dapat mencakup kolom-kolom tertentu atau menyertakan kriteria tertentu dalam pemilihan data. Hal ini memungkinkan pengguna untuk menyesuaikan tampilan data sesuai dengan kebutuhan mereka tanpa mengganggu struktur data asli dalam database. Dengan demikian, implementasi view tidak hanya meningkatkan efisiensi dalam pengambilan keputusan, tetapi juga memungkinkan pengguna untuk menyesuaikan tampilan data sesuai dengan kebutuhan spesifik mereka.

## **Kesimpulan**

1. View adalah tabel virtual yang terbuat dari suatu query terhadap satu tabel atau beberapa tabel yang digunakan untuk menyederhanakan dan mempermudah persepsi pengguna dalam database.
2. Jika sistem database mampu menentukan pemetaan balik dari skema view ke skema tabel dasar, maka view memungkinkan untuk di update. Dalam kondisi ini, operasi-operasi INSERT, UPDATE dan DELETE dapat diterapkan pada view.
3. View dapat dihapus jika sudah tidak dibutuhkan lagi dengan cara menggunakan perintah DROP VIEW view\_name.