**LAPORAN RESMI**

**MODUL V**

**TYPE DATA/STORED PROCEDURE**



**NAMA : ANISYAFAAH**

**N.R.P : 220441100105**

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

**ASISTEN : AFFAN MAULANA ZULKARNAIN**

**TGL PRAKTIKUM : 08 MEI 2024**

**Disetujui : 14 Mei 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**

Di era digital yang semakin berkembang pesat ini, pemahaman tentang tipe data dan penggunaan stored procedure dalam basis data menjadi semakin penting. Pertama-tama, pemahaman yang kuat tentang tipe data memungkinkan kita untuk mengelola informasi dengan lebih efisien dan efektif. Dengan memahami karakteristik setiap tipe data, seperti string, integer, atau float, kita dapat mengoptimalkan struktur basis data untuk menyimpan informasi dengan benar, menghindari pemborosan ruang penyimpanan, dan meningkatkan kinerja sistem secara keseluruhan.

Kedua, penggunaan stored procedure menjadi kunci dalam meningkatkan kinerja dan keamanan basis data. Stored procedure adalah sekumpulan pernyataan SQL yang telah diproses dan disimpan dalam basis data untuk dieksekusi ulang. Dengan menggunakan stored procedure, kita dapat menghindari pengulangan kode, mengurangi beban jaringan dengan mengirimkan hanya panggilan stored procedure, serta meningkatkan keamanan data dengan mengontrol akses langsung ke tabel-tabel basis data.

Terakhir, dalam era di mana data menjadi aset utama bagi banyak organisasi, pemahaman tentang tipe data dan penggunaan stored procedure menjadi landasan untuk inovasi dan pengembangan aplikasi yang handal. Dengan menerapkan prinsip-prinsip ini, para pengembang dapat membuat aplikasi yang lebih skalabel, mudah dielola, dan dapat diandalkan, sehingga memungkinkan organisasi untuk memanfaatkan potensi penuh dari sumber daya informasi yang mereka miliki.

## **Tujuan**

* Mampu memahami dan membuat stored procedure pada basis data
* Mampu menggunakan tipe dta sesuai kebutuhan pada stored procedure

# **BAB II DASAR TEORI**

## **Tipe Data**

Dalam Database Data Type adalah suatu fungsi (function) yang digunakan untuk mengidentifikasi batasan suatu kolom dalam menyimpan dan penulisan format suatu data atau konten tertentu. Penggunaan typedata pada database memiliki beberapa fungsi yaitu: Untuk memberikan batasan atau format pada kolom table suatu database.

Ada lima jenis tipe data sesuai dengan SQL - ANSI 1993 yaitu character string, numeric, temporal, binary, dan boolean.

### **Character String**

Atribut seperti nama dan alamat direpresentasikan oleh character string. Ada 2 macam tipe data untuk merepresentasikan character string, yaitu:

* CHARACTER(<panjang>)

CHAR(<panjang>) menspesifikasikan karakter dengan panjang yang tetap. Sisa karakter yang tidak terpakai umumnya digantikan oleh padding characters (spasi).

* CHARACTER VARYING(<panjang>)

Atau VARCHAR(<panjang>) menspesifikasikan karakter dengan panjang yang fleksibel dan maksimum sesuai dengan <panjang>.

Sintak :

**Type Data [(M)]**

Misal**:**

**CHAR [(M)]**

**VARCHAR [(M)]**

**CREATE TABLE** contoh\_cha (cha **CHAR**(5), varcha **VARCHAR**(5)); **INSERT INTO** contoh\_cha **values** ('a ','a ');

**INSERT INTO** contoh\_cha **values** ('dunia','dunia');

**INSERT INTO** contoh\_cha **VALUES** ('basisdata','basisdata');

### **Numeric**

Data-data seperti usia dan gaji disimpan dalam bentuk angka. Penyimpanan dalam bentuk angka menggunakan tipe data numeric. Ada empat macam tipe data numeric, yaitu:



Sintak:

**Type Data [(M[,D])] [UNSIGNED] [ZEROFILL]**

Misal :

**INT/DECIMAL/FLOAT/DOUBLE[(M)] [UNSIGNED] [ZEROFILL]**

Keterangan :

* **M** : menunjukkan lebar karakter maksimum, jumlah digit keseluruhan
* **D** : jumlah digit dibelakang koma
* **tanda [ dan ]** berarti pemakaiannya adalah optional

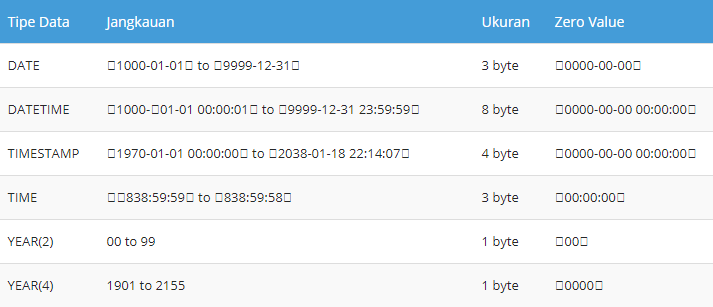
Contoh :

**CREATE TABLE** contoh\_int (mini **TINYINT**, kecil **SMALLINT UNSIGNED**,sedang **MEDIUMINT**(4) **ZEROFILL**, biasa **INT**(4) **UNSIGNED**,besar **BIGINT**(6) **UNSIGNED ZEROFILL**);

### **Menghapus Stored Procedure:**

Temporal merupakan tipe data yang menyimpan tanggal dan waktu yang disesuikan dengan system-timezone (komputer). Sebagai contoh data temporal adalah data tentang tanggal lahir. Ada dua macam tipe data temporal, yaitu:

* + - * DATETIME. Tipe data ini menyimpan informasi tanggal, waktu atau bahkan keduanya. Dalam SQL Server, tipe data ini menyimpan dengan tingkat akurasi sampai 3,33 milidetik. Sedangkan untuk SMALLDATETIME hanya sampai 1 menit. Dalam tipe data ini, juga terdapat tipe data TIMESTAMP dengan tingkat akurasi sampai dengan 9 digit.
      * INTERVAL. Umumnya digunakan untuk menyimpan periode seperti garansi. Ada 2 macam yaitu (1) YEAR-MONTH dan (2) DAY-TIME. SQL Server tidak mempunyai tipe data ini.



Sintak : type data;

Contohnya :

**CREATE TABLE** contoh\_date (dat **DATE**, tim **TIME**, dattim

**DATETIME**,timestam **TIMESTAMP**, yea **YEAR**);

Atau

**SELECT NOW**(), **CURDATE**(), **CURTIME**();



## **Sintaks Stored Procedure**

<create procedure statement> ::=

CREATE PROCEDURE <procedure name> ( [ <parameter list>

] )

<routine body>

<parameter list> ::=

<parameter specification>

[ , <parameter specification> ]...

<parameter specification> ::=

[ IN | OUT | INOUT ] <parameter> <data type>

<routine body> ::= <begin-end block>

<begin-end block> ::=

[ <label> : ] BEGIN <statement list> END [ <label> ]

<statement list> ::= { <body statement> ; }...

<statement in body> ::=

<declarative statement> | <procedural statement>

### **Aktivasi/Pemanggilan Stored Procedure**

<call statement> ::=

CALL [ <database name> . ] <stored procedure name>

( [ <scalar expression> [ , <scalar expression> ]... ] )

### **Menghapus Stored Procedure**

<drop procedure statement> ::

= DROP PROCEDURE [ IF EXISTS ]

[ <database name> . ] <procedure name>

Pernyataan pembuatan stored procedure :

DELIMITER $$

**CREATE PROCEDURE** set\_counter(**INOUT count INT**(4),**IN** inc **INT**(4)) **BEGIN**

**SET count** = **count** + inc;

**END**$$ DELIMITER ;

Cara memanggilnya dengan mengset isi awal :

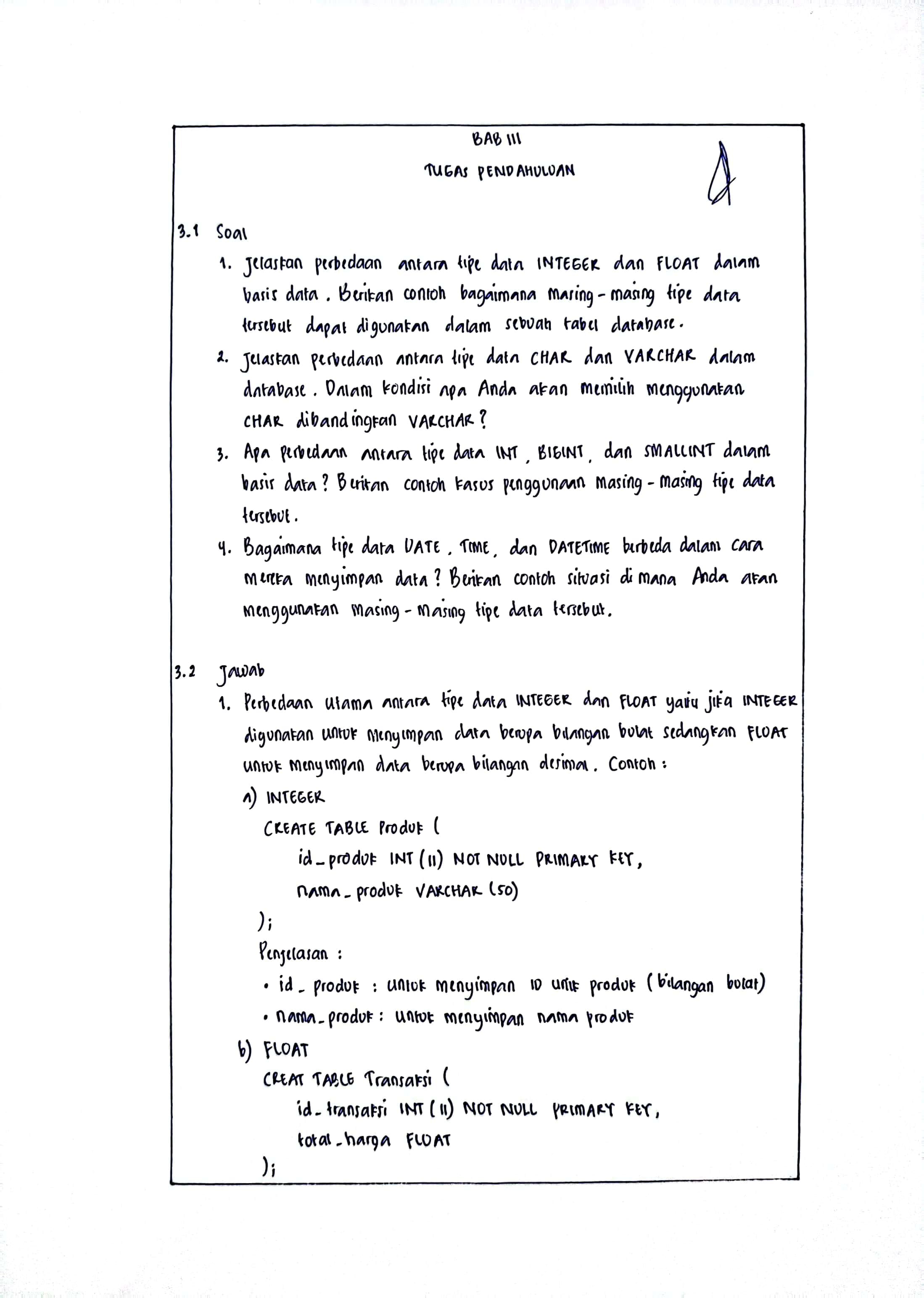
**SET** @counter = 1;

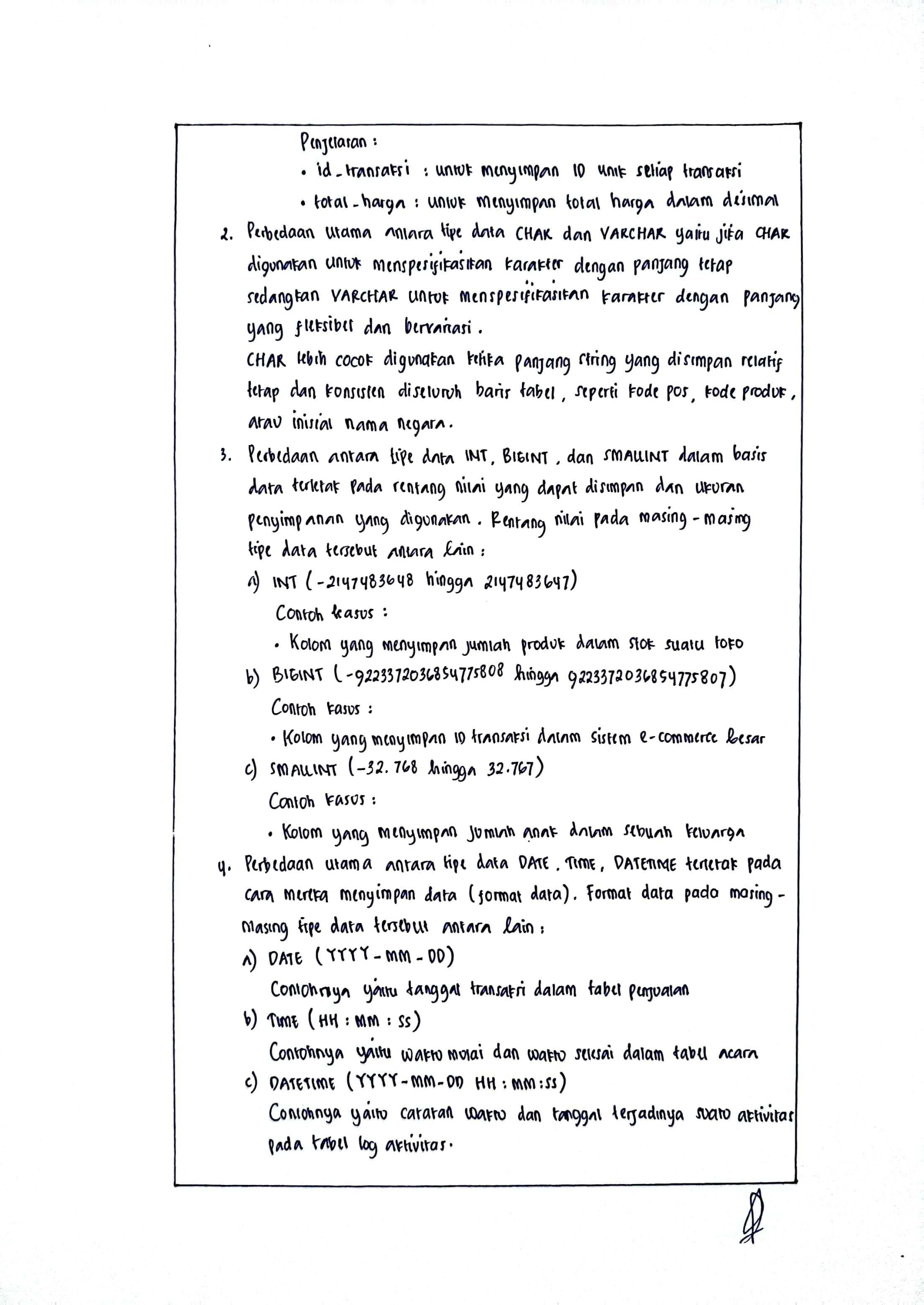
**CALL** set\_counter(@counter,1); *-- 2*

**CALL** set\_counter(@counter,2); *-- 3*

**CALL** set\_counter(@counter,9); *-- 12*

**SELECT** @counter; *-- 12*



****

# **BAB IV IMPLEMENTASI**

## **Source Code**

### **Mengisi Data Setiap Tabel**

* **Tabel Produk**

1. **Source Code**

|  |
| --- |
| CREATE DATABASE Logistik\_Pakaian;  USE Logistik\_Pakaian;  CREATE TABLE Produk (  ID\_Produk INT(11) NOT NULL PRIMARY KEY,  Nama\_Produk VARCHAR(100),  Kategori\_Produk VARCHAR(50),  Harga DOUBLE,  Berat FLOAT  );  INSERT INTO Produk (ID\_Produk, Nama\_Produk, Kategori\_Produk, Harga, Berat) VALUES  (1, 'Kemeja Putih Polos', 'Kemeja', 150000, 0.3),  (2, 'Celana Jeans Slim Fit', 'Celana', 250000, 0.5),  (3, 'Gaun Floral Maxi', 'Gaun', 350000, 0.7),  (4, 'Kaos Oblong Basic', 'Kaos', 80000, 0.2),  (5, 'Jaket Denim Washed', 'Jaket', 300000, 0.6),  (6, 'Rok Midi A-Line', 'Rok', 120000, 0.4),  (7, 'Hoodie Sweater', 'Sweater', 180000, 0.5),  (8, 'Kemeja Tartan Flannel', 'Kemeja', 200000, 0.4),  (9, 'Celana Panjang Chino', 'Celana', 180000, 0.4),  (10, 'Blazer Formal', 'Blazer', 400000, 0.8); |

1. **Penjelasan**

Kode di atas digunakan untuk membuat database Logistik\_Pakaian dan membuat tabel produk serta mengisinya dengan 10 data. Untuk membuat database logistik pakaian menggunakan perintah CREATE DATABASE Logistik\_Pakaian. 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, kategori produk, harga, dan berat. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

* **Tabel Supplier**

1. **Source Code**

|  |
| --- |
| CREATE TABLE Supplier (  ID\_Supplier INT(11) NOT NULL PRIMARY KEY,  Nama\_Supplier VARCHAR(100),  Alamat VARCHAR(255),  Telepon CHAR(15),  Email VARCHAR(100)  );  INSERT INTO Supplier (ID\_Supplier, Nama\_Supplier, Alamat, Telepon, Email) VALUES  (1, 'PT. Amanah Textile', 'Jl. Pahlawan No. 123, Jakarta', '021-12345678', 'info@amanah-textile.com'),  (2, 'CV. Maju Jaya Garment', 'Jl. Raya Industri No. 45, Bandung', '022-98765432', 'info@maju-jaya-garment.com'),  (3, 'UD. Bersama Fashion', 'Jl. Merdeka No. 87, Surabaya', '031-56789012', 'info@bersama-fashion.co.id'),  (4, 'PT. Sentosa Apparel', 'Jl. Pelangi No. 55, Semarang', '024-34567890', 'info@sentosa-apparel.com'),  (5, 'CV. Makmur Abadi Textile', 'Jl. Jaya No. 10, Solo', '0271-2345678', 'info@makmur-abadi-textile.co.id'),  (6, 'UD. Sejahtera Garment', 'Jl. Damai No. 20, Medan', '061-7890123', 'info@sejahtera-garment.com'),  (7, 'PT. Lancar Jaya Fashion', 'Jl. Harmoni No. 30, Malang', '0341-8901234', 'info@lancar-jaya-fashion.com'),  (8, 'CV. Bahagia Textile', 'Jl. Bahagia No. 5, Makassar', '0411-5678901', 'info@bahagia-textile.co.id'),  (9, 'PT. Jaya Bersama Apparel', 'Jl. Bersama No. 3, Palembang', '0711-2345678', 'info@jaya-bersama-apparel.com'),  (10, 'UD. Sukses Fashion', 'Jl. Sukses No. 8, Yogyakarta', '0274-7890123', 'info@sukses-fashion.co.id'); |

1. **Penjelasan**

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

* **Tabel Gudang**

1. **Source Code**

|  |
| --- |
| CREATE TABLE Gudang (  ID\_Gudang INT(11) NOT NULL PRIMARY KEY,  Nama VARCHAR(100),  Alamat VARCHAR(255)  );  INSERT INTO Gudang (ID\_Gudang, Nama, Alamat) VALUES  (1, 'Gudang Utama', 'Jl. Gatot Subroto No. 123, Jakarta'),  (2, 'Gudang Pusat', 'Jl. Sudirman No. 45, Bandung'),  (3, 'Gudang Sentral', 'Jl. Diponegoro No. 87, Surabaya'),  (4, 'Gudang Logistik', 'Jl. Gajah Mada No. 55, Semarang'),  (5, 'Gudang Distribusi', 'Jl. Pemuda No. 10, Solo'),  (6, 'Gudang Amanah', 'Jl. Merdeka No. 20, Medan'),  (7, 'Gudang Maju', 'Jl. Dipa No. 30, Malang'),  (8, 'Gudang Bersama', 'Jl. Hasanuddin No. 5, Makassar'),  (9, 'Gudang Sejahtera', 'Jl. Kapten No. 3, Palembang'),  (10, 'Gudang Lancar', 'Jl. Pahlawan No. 8, Yogyakarta'); |

1. **Penjelasan**

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

* **Tabel Stok**

1. **Source Code**

|  |
| --- |
| CREATE TABLE Stok (  ID\_Stok INT(11) NOT NULL PRIMARY KEY,  ID\_Produk INT(11),  ID\_Gudang INT(11),  Jumlah INT(11),  Tanggal\_Update DATETIME,  FOREIGN KEY (ID\_Produk) REFERENCES Produk (ID\_Produk),  FOREIGN KEY (ID\_Gudang) REFERENCES Gudang (ID\_Gudang)  );  INSERT INTO Stok (ID\_Stok, ID\_Produk, ID\_Gudang, Jumlah, Tanggal\_Update) VALUES  (1, 1, 1, 100, '2024-05-01 09:00:00'),  (2, 2, 2, 80, '2024-05-02 09:15:00'),  (3, 3, 3, 50, '2024-05-03 09:30:00'),  (4, 4, 4, 120, '2024-05-04 09:45:00'),  (5, 5, 5, 70, '2024-05-05 10:00:00'),  (6, 6, 6, 90, '2024-05-06 10:15:00'),  (7, 7, 7, 60, '2024-05-07 10:30:00'),  (8, 8, 8, 110, '2024-05-08 10:45:00'),  (9, 9, 9, 85, '2024-05-09 11:00:00'),  (10, 10, 10, 40, '2024-05-10 11:15:00'); |

1. **Penjelasan**

Kode di atas digunakan untuk membuat tabel stok dan mengisi data sebanyak 10 data. Untuk membuat tabel stok menggunakan perintah CREATE TABLE Stok (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id stok sebagai primary key, id produk yang terhubung dengan tabel produk menggunakan perintah FOREIGN KEY, id gudang yang terhubung dengan tabel gudang menggunakan perintah FOREIGN KEY, jumlah, dan tanggal update. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

* **Tabel Karyawan**

1. **Source Code**

|  |
| --- |
| CREATE TABLE Karyawan (  ID\_Karyawan INT(11) NOT NULL PRIMARY KEY,  ID\_Gudang INT(11),  Nama VARCHAR(100),  Alamat VARCHAR(255),  Posisi VARCHAR(50),  Gaji DOUBLE,  FOREIGN KEY (ID\_Gudang) REFERENCES Gudang (ID\_Gudang)  );  INSERT INTO Karyawan (ID\_Karyawan, ID\_Gudang, Nama, Alamat, Posisi, Gaji) VALUES  (1, 1, 'Budi Santoso', 'Jl. Merdeka No. 1, Jakarta', 'Manajer Gudang', 8000000),  (2, 2, 'Ani Wijaya', 'Jl. Sudirman No. 2, Bandung', 'Staf Gudang', 5000000),  (3, 3, 'Cahyo Nugroho', 'Jl. Diponegoro No. 3, Surabaya', 'Staf Gudang', 5000000),  (4, 4, 'Dewi Kurniawan', 'Jl. Gajah Mada No. 4, Semarang', 'Staf Gudang', 5000000),  (5, 5, 'Eka Setiawan', 'Jl. Pemuda No. 5, Solo', 'Staf Gudang', 5000000),  (6, 6, 'Fita Dewanti', 'Jl. Merdeka No. 6, Medan', 'Staf Gudang', 5000000),  (7, 7, 'Galih Susanto', 'Jl. Dipa No. 7, Malang', 'Staf Gudang', 5000000),  (8, 8, 'Hani Maulana', 'Jl. Hasanuddin No. 8, Makassar', 'Staf Gudang', 5000000),  (9, 9, 'Indra Wibowo', 'Jl. Kapten No. 9, Palembang', 'Staf Gudang', 5000000),  (10, 10, 'Joko Prasetyo', 'Jl. Pahlawan No. 10, Yogyakarta', 'Staf Gudang', 5000000); |

1. **Penjelasan**

Kode di atas digunakan untuk membuat tabel karyawan dan mengisi data sebanyak 10 data. Untuk membuat tabel karyawan menggunakan perintah CREATE TABLE Karyawan (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id karyawan sebagai primary key, id gudang yang terhubung dengan tabel gudang menggunakan perintah FOREIGN KEY, nama karyawan, alamat, posisi, dan gaji. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

* **Tabel Transaksi**

1. **Source Code**

|  |
| --- |
| CREATE TABLE Transaksi (  ID\_Transaksi INT(11) NOT NULL PRIMARY KEY,  ID\_Produk INT(11),  ID\_Supplier INT(11),  ID\_Karyawan INT(11),  Jumlah INT(11),  Total\_Harga DOUBLE,  Tanggal\_Transaksi DATETIME,  FOREIGN KEY (ID\_Produk) REFERENCES Produk (ID\_Produk),  FOREIGN KEY (ID\_Supplier) REFERENCES Supplier (ID\_Supplier),  FOREIGN KEY (ID\_Karyawan) REFERENCES Karyawan (ID\_Karyawan)  );  INSERT INTO Transaksi (ID\_Transaksi, ID\_Produk, ID\_Supplier, ID\_Karyawan, Jumlah, Total\_Harga, Tanggal\_Transaksi) VALUES  (1, 1, 1, 1, 3, 180000, '2024-05-03 09:00:00'),  (2, 2, 2, 2, 4, 190000, '2024-05-04 09:15:00'),  (3, 3, 3, 3, 30, 10500000, '2024-05-05 09:30:00'),  (4, 4, 4, 4, 60, 4800000, '2024-05-06 09:45:00'),  (5, 5, 5, 5, 45, 13500000, '2024-05-07 10:00:00'),  (6, 6, 6, 6, 55, 6600000, '2024-05-08 10:15:00'),  (7, 7, 7, 7, 35, 6300000, '2024-05-09 10:30:00'),  (8, 8, 8, 8, 70, 14000000, '2024-05-10 10:45:00'),  (9, 9, 9, 9, 65, 11700000, '2024-05-11 11:00:00'),  (10, 10, 10, 10, 25, 10000000, '2024-05-12 11:15:00'); |

1. **Penjelasan**

Kode di atas digunakan untuk membuat tabel transaksi dan mengisi data sebanyak 10 data. Untuk membuat tabel transaksi menggunakan perintah CREATE TABLE Transaksi (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id transaksi sebagai primary key, id produk yang terhubung dengan tabel produk menggunakan perintah FOREIGN KEY, id supplier yang terhubung dengan tabel supplier menggunakan perintah FOREIGN KEY, id karyawan yang terhubung dengan tabel karyawan menggunakan perintah FOREIGN KEY, jumlah, total harga, dan tanggal transaksi. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

### **Stored Procedure Menghtiung Total Harga Berdasarkan Karyawan**

1. **Source Code**

|  |
| --- |
| -- Nomor 1  DELIMITER //  CREATE PROCEDURE TotalTransaksi (  IN Karyawan INT,  OUT TotalHarga DOUBLE  )  BEGIN  SELECT SUM(Total\_Harga) INTO TotalHarga  FROM Transaksi  WHERE ID\_Karyawan = Karyawan;  END//  DELIMITER ;  CALL TotalTransaksi(2, @TotalHarga);  SELECT @TotalHarga; |

1. **Penjelasan**

Kode di atas digunakan untuk membuat sebuah prosedur yang menggunakan parameter IN dan OUT. Kode di atas akan menampilkan jumlah total harga keseluruhan berdasarkan id karyawan menggunakan fungsi agregasi (SUM). Untuk membuat prosedur menggunakan perintah CREATE PROCEDURE nama\_prosedur. IN menjadi parameter pemanggil dimana data akan dicari berdasarkan id karyawan dan OUT menjadi parameter untuk menyimpan nilai dari fungsi agregasi sehingga kode eksekusi akan dimulai dari perintah SELECT SUM(Total\_Harga) INTO nama parameter OUT FROM Transaksi dimana ID\_Karyawan = nama parameter IN. Untuk menampilkan datanya menggunakan syntax CALL nama\_prosedur (id\_karyawan yang dicari, @nama parameter OUT. @nama parameter adalah sebuah variabel yang akan diisi dengan nilai yang dikirimkan saat prosedur dipanggil.

### **Stored Procedure Lama Setiap Produk di Gudang Sejak Tanggal Masuk**

1. **Source Code**

|  |
| --- |
| -- Nomor 2  DELIMITER //  CREATE PROCEDURE LamaProduk()  BEGIN  SELECT P.nama\_produk, G.nama, DATEDIFF(CURDATE(), S.tanggal\_update) AS Lama\_Di\_Gudang  FROM Produk P  JOIN Stok S ON P.ID\_Produk = S.ID\_Produk  JOIN Gudang G ON S.ID\_Gudang = G.ID\_Gudang;  END //  DELIMITER ;  CALL LamaProduk(); |

1. **Penjelasan**

Kode di atas digunakan untuk menampilkan data lama dalam hari setiap produk berada di gudang selama tanggal masuk. Untuk membuat prosedur menggunakan perintah CREATE PROCEDURE nama\_prosedur. Prosedur ini tidak memiliki parameter, sehingga akan menjadi sebuah fungsi saja tanpa menggunakan parameter. Kode akan dimulai dari perintah untuk menampilkan nama produk, nama gudang, dan kode perhitungan selisih dalam jumlah hari (DATEDIFF) antara tanggal saat ini (CURDATE()) dan tanggal terakhir update dari tabel stok. Kode untuk menampilkan seluruh komponen tersebut berasal dari tabel stok dan gudang menggunakan perintah JOIN. Untuk menampilkan datanya dapat menggunakan syntax CALL nama\_prosedur.

### **Stored Procedure Menghapus Transaksi yang Kurang Dari 200 Ribu Pada Satu Bulan Terakhir**

1. **Source Code**

|  |
| --- |
| -- Nomor 3  DELIMITER //  CREATE PROCEDURE Hapus\_Transaksi()  BEGIN  DELETE FROM Transaksi  WHERE Tanggal\_Transaksi >= DATE\_SUB(CURDATE(), INTERVAL 1 MONTH)  AND Total\_Harga <= 200000;  END //  DELIMITER ;  CALL Hapus\_Transaksi(); |

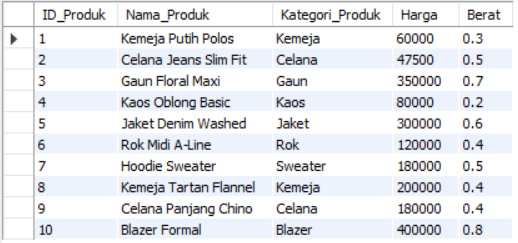
1. **Penjelasan**

Kode di atas digunakan untuk menghapus data pada tabel transaksi yang memiliki total harga kurang dari 200000 dalam sebulan terakhir. Untuk membuat prosedur menggunakan perintah CREATE PROCEDURE nama\_prosedur. Prosedur ini tidak memiliki parameter, sehingga akan menjadi sebuah fungsi saja tanpa menggunakan parameter. Kode akan dimulai dari perintah DELETE dari tabel Transaksi dimana (WHERE) tanggal transaksinya yang dilakukan dalam satu bulan terakhir akan dihapus menggunakan fungsi DATE\_SUB dan total harga diatur kurang dari 200000. Untuk menjalankan prosedur dapat menggunakan syntax CALL nama\_prosedur.

## **Hasil**

### **Data Setiap Tabel**

* **Tabel Produk**



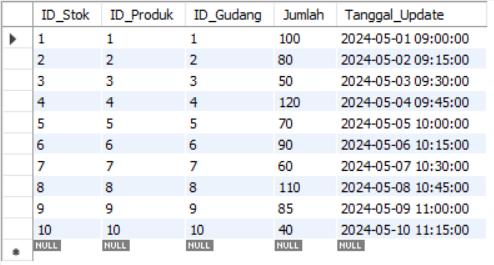
* **Tabel Supplier**



* **Tabel Gudang**



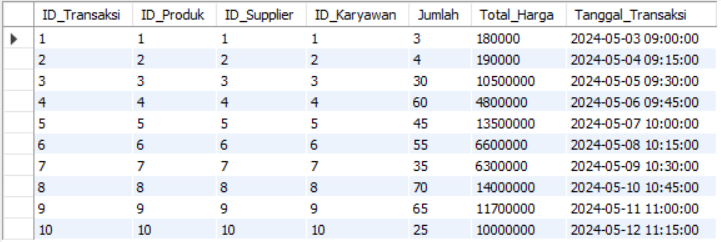
* **Tabel Stok**



* **Tabel Karyawan**



* **Tabel Transaksi**



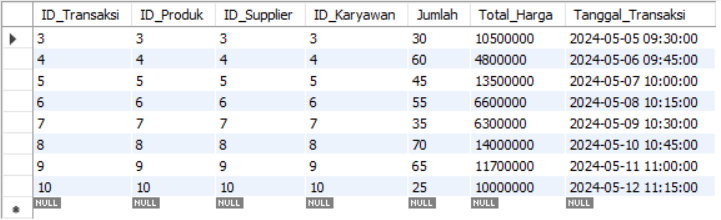
### **Stored Procedure Menghtiung Total Harga Berdasarkan Karyawan**



### **Stored Procedure Lama Setiap Produk di Gudang Sejak Tanggal Masuk**



### **Stored Procedure Menghapus Transaksi yang Kurang Dari 200 Ribu Pada Satu Bulan Terakhir**



# **BAB V PENUTUP**

## **Analisa**

Dari hasil praktikum, praktikan menganalisa bahwa penggunaan tipe data yang tepat dalam sistem manajemen basis data (DBMS) adalah kunci untuk memastikan keakuratan dan konsistensi data. Dengan memilih tipe data yang sesuai, seperti string untuk teks, integer untuk angka bulat, atau float untuk angka desimal, DBMS dapat mengelola data dengan efisien dan mengoptimalkan penggunaan penyimpanan. Misalnya, menggunakan tipe data yang lebih kecil untuk kolom-kolom yang membutuhkan kapasitas penyimpanan yang lebih kecil dapat menghemat ruang penyimpanan dan meningkatkan kinerja sistem.

Penerapan stored procedure dalam sistem manajemen basis data memberikan sejumlah manfaat yang signifikan. Stored procedure memungkinkan pengguna untuk mengelompokkan serangkaian perintah SQL menjadi satu unit logis, yang dapat dipanggil dan dieksekusi secara berulang tanpa perlu menulis ulang kode. Selain itu, stored procedure juga membantu meningkatkan keamanan data dengan mengizinkan kontrol akses yang ketat. Dengan menyimpan logika bisnis di dalam stored procedure, pengguna dapat mengontrol akses langsung ke tabel-tabel basis data, sehingga mengurangi risiko eksploitasi dan kebocoran data yang tidak diinginkan.

## **Kesimpulan**

1. Tipe data merupakan bagian dari variabel yang mempengaruhi perilaku variabel. Dengan tipe data ini, bisa ditentukan nilai apa yang bisa disimpan didalam variabel tersebut.
2. Stored Procedure adalah sebuah prosedur layaknya subprogram (subrutin) di dalam bahasa pemrograman reguler yang tersimpan di dalam katalog basis data.
3. Dengan demikian, penggunaan tipe data yang tepat dan stored procedure dalam basis data tidak hanya meningkatkan efisiensi operasional, tetapi juga meningkatkan keamanan dan keandalan sistem secara keseluruhan.
4. Top of Form