**LAPORAN PRAKTIKUM 4**

**BASIS DATA**

Disusun untuk Memenuhi Matakuliah Praktikum Basis Data

Dibimbing oleh Sulaibatul Aslamiyah, M.Kom



**Oleh:**

Oleh

**Rachmad Lukman Maulana**

**1124102171**

# PROGRAM STUDI S1 TEKNIK INFORMATIKA

**SEKOLAH TINGGI ILMU KOMPUTER PGRI BANYUWANGI**

**2025**

**LEMBAR PENGESAHAN**

# LAPORAN PRAKTIKUM

Matakuliah : Praktikum Database

Oleh : Rachmad Lukman Maulana

NIM : 1124102171

**Telah disahkan pada**

Hari : Kamis

Tanggal : 24 july 2025

**Mengetahui/ Menyetujui :**

Dosen Pengampu Mata Kuliah

**Sulaibatul Aslamiyah S.Kom M.Kom**

NIDN. 0712018304

**KATA PENGANTAR**

Puji syukur kita panjatkan kepada Tuhan Yang Maha Esa, karena berkat rahmat-Nya, penyusunan materi mata kuliah *Database MySQL* ini dapat diselesaikan dengan baik. Mata kuliah *Database MySQL* merupakan bagian penting dalam pengembangan kemampuan mahasiswa dalam bidang teknologi informasi dan komputer. Dalam mata kuliah ini, mahasiswa akan mempelajari dasar-dasar tentang pengelolaan dan manajemen data menggunakan sistem manajemen basis data MySQL yang banyak digunakan dalam pengembangan aplikasi berbasis web maupun desktop.

Database adalah komponen penting dalam sistem informasi modern, dan MySQL sebagai salah satu sistem manajemen basis data relasional yang paling populer, memberikan landasan yang kuat bagi mahasiswa untuk mengelola dan mengorganisir data secara efisien. Materi yang akan dipelajari mencakup struktur dasar database, perancangan tabel, pengolahan data, hingga query yang digunakan untuk memanipulasi dan mengambil data yang relevan sesuai kebutuhan.

Dengan mempelajari mata kuliah ini, diharapkan mahasiswa dapat memahami konsep-konsep dasar yang akan berguna dalam dunia profesional, serta mampu mengembangkan dan menerapkan solusi berbasis database yang efektif dan efisien. Selain itu, penguasaan terhadap MySQL juga akan memperluas keterampilan mahasiswa dalam menghadapi tantangan dunia teknologi yang semakin berkembang pesat.

Semoga materi yang disampaikan dalam mata kuliah ini dapat memberikan manfaat yang sebesar-besarnya bagi mahasiswa dan dapat diterapkan dengan baik dalam dunia kerja maupun dalam pengembangan karir di masa depan. Terima kasih kepada semua pihak yang telah mendukung terselenggaranya mata kuliah ini, baik dari pengajar, staf pengelola, maupun mahasiswa yang telah berpartisipasi dengan antusias.

**MODUL PRAKTIKUM 4**

**1.1 Capaian Praktikum Pertemuan 4**

• Mahasiswa mempelajari tentang trigger, subquery dan store procedure

## 1.2 Indikator Capaian

* Mahasiswa mempelajari perintah seperti Trigger, Subselect, serta penyeleksian.
* Mahasiswa dapat mengolah data pada suatu tabel berdasarkan perintahperintah yang tertera.

## 1.3 Landasan Teori

* Subquery atau SubSelect adalah query yang ditanam di dalam query utama untuk melakukan seleksi atau pengolahan data kompleks. Ini memungkinkan pemilihan data berdasarkan hasil query internal, yang digunakan sebagai filter atau kondisi di dalam query utama.

* Trigger adalah objek database yang aktif otomatis saat terjadi perubahan data tertentu dalam tabel terkait. Digunakan untuk menjalankan tindakan atau prosedur khusus saat terjadi operasi seperti INSERT, UPDATE, atau DELETE pada tabel.

* Seleksi dengan kondisi adalah proses memilih data dari tabel berdasarkan kriteria tertentu yang ditentukan menggunakan pernyataan WHERE. Ini memungkinkan filter data untuk mengambil hanya data yang memenuhi syarat, memungkinkan manipulasi data yang lebih tepat dan efisien.

## 1.4 Pelaksanaan Praktikum

### 1.4.1 Percobaan Pertama

Berikut ini contoh trigger yang mencatat aktivitas ke tabel log setiap kali terjadi proses insert ke tabel pelanggan. Sebelumnya, tabel log dibuat untuk menyimpan hasil triggering tersebut.

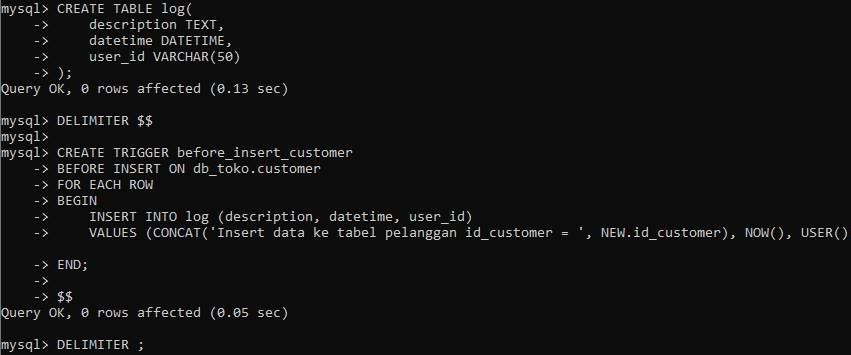
## a. Script / Setting Program

|  |
| --- |
| mysql> CREATE TABLE log(  -> description TEXT,  -> datetime DATETIME,  -> user\_id VARCHAR(50)  -> );  Query OK, 0 rows affected (0.13 sec) mysql> DELIMITER  $$ mysql> CREATE TRIGGER before\_insert\_customer  -> BEFORE INSERT ON db\_toko.customer  -> FOR EACH ROW  -> BEGIN  -> INSERT INTO log (description, datetime, user\_id)  -> VALUES (CONCAT('Insert data ke tabel pelanggan id\_customer = ', NEW.id\_customer), NOW(), USER());  -> END;  -> $$  Query OK, 0 rows affected (0.05 sec) mysql> DELIMITER  ; |

## b. Langkah Uji Coba

1. Membuat tabel log yang akan digunakan untuk aktivitas logSalin.
2. Membuat trigger `before\_insert\_customer` di database `db\_toko`. Setiap kali data baru dimasukkan ke tabel `customer`, trigger ini akan memasukkan catatan ke tabel `log`. Catatan tersebut berisi deskripsi kegiatan, waktu saat ini, dan ID pengguna yang melakukan operasi.

## c. Hasil Uji Coba

 **d.**

## Analisa Hasil

1. Berhasil membuat tabel log dengan kolom-kolom description, datetime, dan user\_id.
2. Berhasil membuat Trigger before\_insert\_customer dan terkait dengan tabel customer di dalam database db\_toko.Output:
3. Tabel log sudah dibuat. Kolom description berisi deskripsi kegiatan, datetime untuk waktu pencatatan, dan user\_id untuk ID pengguna yang melakukan operasi.
4. Trigger `before\_insert\_customer` sudah dibuat dan diuji. Ketika data baru ditambahkan ke tabel customer, trigger secara otomatis mencatat aktivitas tersebut ke dalam tabel log.

### 1.4.2 Percobaan Kedua

Stored Procedure terbagi menjadi dua jenis utama: FUNCTION dan PROCEDURE. Perbedaan utama antara keduanya adalah bahwa FUNCTION mengembalikan nilai (return value), sedangkan PROCEDURE tidak mengembalikan nilai. PROCEDURE umumnya berisi serangkaian proses yang dapat menampilkan output tetapi tidak menghasilkan nilai yang dikembalikan.

## a. Script / Setting Program

mysql> DELIMITER $$ mysql> mysql>

CREATE PROCEDURE jumlahCustomer()

-> BEGIN

-> SELECT COUNT(\*) FROM customer;

-> END$$

Query OK, 0 rows affected (0.04 sec) mysql> mysql> DELIMITER ; mysql>

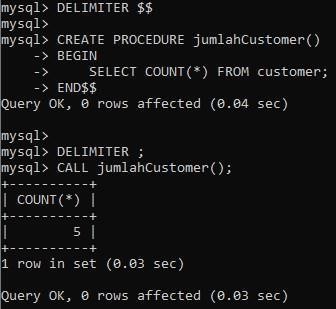
CALL jumlahCustomer();

## b. Langkah Uji Coba

1. Membuat Stored Procedure `jumlahCustomer` untuk menghitung jumlah baris (record) dalam tabel customer.
2. Memanggil Stored Procedure `jumlahCustomer` untuk menghitung jumlah total record dalam tabel customer..

## c. Hasil Uji Coba

1. Stored procedure `jumlahCustomer` berhasil dibuat tanpa adanya masalah.
2. Saat dipanggil dengan perintah `CALL jumlahCustomer();`, MySQL akan menjalankan stored procedure tersebut dan mengembalikan jumlah total record dalam tabel customer.



## d. Analisa Hasil

1. Stored procedure jumlahCustomer berhasil dibuat dengan menggunakan perintah SELECT COUNT(\*) FROM customer;. Pendekatan ini sangat efisien karena secara langsung menghitung jumlah baris dalam tabel customer dengan menggunakan fungsi agregat COUNT(\*), yang dirancang khusus agar lebih Efisien dan sederhana.

### 1.4.3 Percobaan Ketiga

Pada bentuk procedure yang kedua di atas “jumlahPelanggan2”, dapat disimpan hasil dari procedure ke dalam satu variabel bernama hasil yang bertipe INT. Perbedaan dari kedua bentuk di atas adalah, pada bentuk kedua, dapat dilakukan pemanggilan procedure dengan SELECT, sedangkan pada yang pertama tidak bisa. Untuk mengubah atau menghapus SP, perintahnya sebagai berikut:

## a. Script / Setting Program

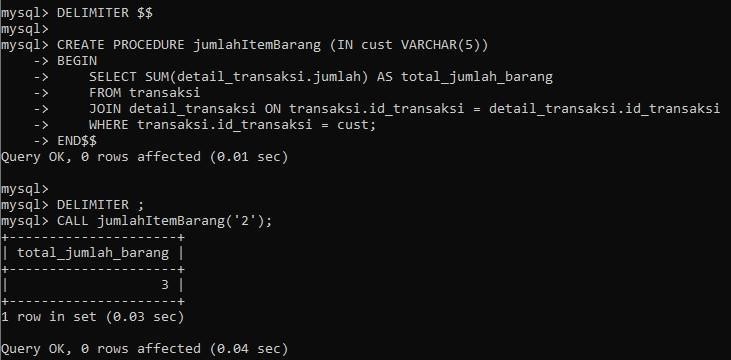
|  |
| --- |
| mysql> DELIMITER $$ mysql> mysql> CREATE PROCEDURE jumlahItemBarang (IN cust VARCHAR(5))  -> BEGIN  -> SELECT SUM(detail\_transaksi.jumlah) AS total\_jumlah\_barang  -> FROM transaksi  -> JOIN detail\_transaksi ON transaksi.id\_transaksi = detail\_transaksi.id\_transaksi  -> WHERE transaksi.id\_transaksi = cust;  -> END$$  Query OK, 0 rows affected (0.01 sec) mysql> mysql> DELIMITER ; mysql>  CALL jumlahItemBarang('2'); |

## b. Langkah Uji Coba

1. Buat terlebih dahulu Stored Procedure “jumlahItemBarang”
2. Lalu memanggil Stored Procedure “jumlahItemBarang”

## c. Hasil Uji Coba

1. Stored procedure “jumlahItemBarang” berhasil dibuat tanpa terkendala masalah
2. Ketika stored procedure dipanggil dengan perintah `CALL jumlahItemBarang('2');`, MySQL akan mengeksekusi prosedur dengan menggunakan ID transaksi '2' sebagai parameter dan mengembalikan jumlah total barang yang dibeli dalam transaksi tersebut sebagai hasil..



## d. Analisa Hasil

1. Stored procedure jumlahItemBarang menghitung jumlah total barang dengan menggabungkan tabel transaksi dan detail\_transaksi, kemudian mengelompokkan berdasarkan ID transaksi yang diberikan.

### 1.4.4 Percobaan Keempat

Dengan adanya penyeleksian kondisi, dapat dilakukan pengaturan alur proses yang terjadi dalam database. Di MySQL, penyeleksian kondisi terdiri dari IF, IF...ELSE dan CASE.

Berikut ini bentuk umum ketiga perintah tersebut:

## a. Script / Setting Program

mysql> DELIMITER ; mysql> SELECT

COUNT(id\_transaksi) AS Total,

-> IF(COUNT(id\_transaksi) > 0,

-> CONCAT('Jumlah transaksi: ', COUNT(id\_transaksi), ' kali'),

-> 'Tidak ada transaksi') AS 'Status'

-> FROM transaksi

-> WHERE id\_customer = 'P001'; mysql> SELECT

COUNT(id\_transaksi) AS 'Total Transaksi', CASE

-> WHEN COUNT(id\_transaksi) = 1 THEN 'Sedikit'

-> WHEN COUNT(id\_transaksi) = 2 THEN 'Lumayan'

-> WHEN COUNT(id\_transaksi) > 3 THEN 'Banyak'

-> ELSE 'Tidak ada transaksi' END AS 'Status'

-> FROM transaksi

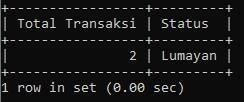
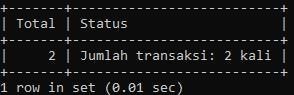
-> WHERE id\_customer='P001';

## b. Langkah Uji Coba

1. Menjalankan program Query untuk Menghitung Jumlah Transaksi dan untuk menentukan Statusnya
2. Menjalankan program Query untuk Menghitung Jumlah Transaksi dan Menentukan Status dengan kode “CASE”

## c. Hasil Uji Coba

1. Query ini menghasilkan dua kolom: Total, yang berisi jumlah total transaksi, dan Status, yang berisi pesan status berdasarkan jumlah transaksi.
2. Query akan menghasilkan dua kolom: Total Transaksi, menunjukkan jumlah total transaksi, dan Status, menampilkan status berdasarkan rentang jumlah transaksi yang ditentukan menggunakan CASE.



## d. Analisa Hasil

1. Query berhasil menghitung jumlah total transaksi dan menampilkan pesan status berdasarkan kondisi yang diberikan.
2. Query berhasil menghitung jumlah total transaksi dan menampilkan status berdasarkan kondisi yang ditentukan dalam CASE.

### 1.4.5 Percobaan Kelima

Tampilkan id dan nama customer dari tabel customer yang pernah melakukan transaksi pada tabel transaksi menggunakan subquery (SELECT dalam SELECT).

Gunakan perintah CASE untuk mengubah field jumlah dari angka menjadi huruf pada detail transaksi dan tabel barang, menampilkan id transaksi, nama barang, dan jumlah, serta mengurutkan data berdasarkan id transaksi secara ascending.

## a. Script / Setting Program

|  |
| --- |
| mysql> SELECT id\_customer, nama\_customer  -> FROM customer  -> WHERE id\_customer IN (  -> SELECT DISTINCT id\_customer  -> FROM transaksi  -> ); mysql> DELIMITER $$ mysql> CREATE  PROCEDURE showDetailTransaksi()  -> BEGIN  -> SELECT dt.id\_transaksi, b.nama\_barang,  -> CASE  -> WHEN dt.jumlah = 1 THEN 'Satu'  -> WHEN dt.jumlah = 2 THEN 'Dua'  -> WHEN dt.jumlah = 3 THEN 'Tiga'  -> ELSE 'Lebih dari tiga'  -> END AS Jumlah  -> FROM detail\_transaksi dt  -> JOIN barang b ON dt.id\_barang = b.id\_barang  -> ORDER BY dt.id\_transaksi ASC;  -> END$$  Query OK, 0 rows affected (0.01 sec) mysql> mysql> DELIMITER ; mysql>  CALL showDetailTransaksi(); |

* Query pertama memilih id\_customer dan nama\_customer dari tabel customer, memfilter hanya customer yang pernah melakukan transaksi berdasarkan id\_customer yang ada di tabel transaksi.

* Stored procedure showDetailTransaksi dibuat untuk menampilkan detail transaksi dengan id\_transaksi, nama\_barang dari tabel barang, dan mengubah nilai jumlah dari angka menjadi huruf berdasarkan kondisi yang ditentukan menggunakan CASE.

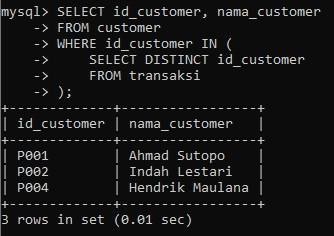
* Stored procedure dipanggil menggunakan perintah CALL showDetailTransaksi(); untuk mengeksekusi dan menampilkan hasil dari query di dalamnya.

## b. Langkah Uji Coba

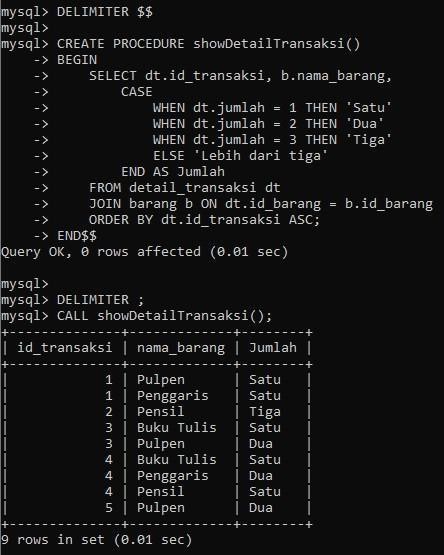
1. Menjalankan query untuk memilih customer yang memiliki transaksi.
2. Membuat stored procedure showDetailTransaksi untuk menampilkan detail transaksi, termasuk mengubah jumlah menjadi huruf berdasarkan kondisi.
3. Memanggil stored procedure showDetailTransaksi untuk eksekusi dan tampilan hasilnya

## c. Hasil Uji Coba

1. Query menghasilkan daftar customer yang memiliki setidaknya satu transaksi beserta nama mereka.



1. Stored procedure mengembalikan daftar detail transaksi dengan ID transaksi, nama barang, dan jumlah barang yang telah diubah menjadi deskripsi teks.



## d. Analisa Hasil

1. Query berhasil memilih customer yang memiliki transaksi dari tabel customer menggunakan subquery untuk memeriksa tabel transaksi.
2. Stored procedure showDetailTransaksi telah berhasil dibuat dan dieksekusi dengan benar, menghasilkan daftar detail transaksi yang mengikuti logika pengelompokan jumlah barang yang telah ditentukan.

### 1.5 Kesimpulan

#### 1.5.1 Kesimpulan Percobaan 1

Pada uji coba pertama, trigger MySQL telah diimplementasikan untuk mencatat setiap operasi insert ke dalam tabel customer. Trigger `before\_insert\_customer` mencatat otomatis kegiatan, waktu pencatatan, dan ID pengguna yang melakukan operasi ke dalam tabel log. Ini meningkatkan transparansi dan audit trail terhadap operasi data dalam database MySQL.

#### 1.5.2 Kesimpulan Percobaan 2

Pada uji coba kedua, berhasil dibuat stored procedure `jumlahCustomer` untuk menghitung jumlah record dalam tabel customer di MySQL. Stored procedure ini memberikan cara yang efisien dan terstruktur untuk mendapatkan informasi tentang jumlah total record dalam tabel tersebut tanpa perlu menulis ulang query. Implementasi ini mempermudah pengelolaan penghitungan data dalam aplikasi atau sistem berbasis MySQL.

#### 1.5.3 Kesimpulan Percobaan 3

Pada uji coba ketiga, berhasil dibuat stored procedure `jumlahItemBarang` yang menggunakan parameter untuk menghitung jumlah total barang dalam sebuah transaksi berdasarkan ID transaksi yang diberikan. Stored procedure ini berjalan dengan baik dan menunjukkan kemampuan MySQL dalam mengelola prosedur dengan parameter untuk memberikan hasil yang dinamis sesuai kebutuhan pengguna. Implementasi ini memberikan fleksibilitas dalam mengakses dan memproses data transaksi dalam database MySQL..

#### 1.5.4 Kesimpulan Percobaan 4

Pada uji coba keempat, menggunakan fungsi kondisional IF dan CASE dalam query MySQL untuk mengevaluasi jumlah transaksi dan menentukan status berdasarkan kondisi tertentu. Implementasi ini berhasil menentukan pesan status dengan IF berdasarkan keberadaan transaksi, dan menggunakan CASE untuk memberikan status berdasarkan rentang jumlah transaksi. Hal ini memungkinkan pengguna untuk dengan mudah menghasilkan informasi status transaksi berdasarkan kondisi yang spesifik, meningkatkan kemampuan analisis data dalam aplikasi atau sistem basis data..

#### 1.5.5 Kesimpulan Percobaan 5

Pada uji coba kelima, digunakan subquery dan stored procedure dalam MySQL untuk menampilkan detail transaksi dengan deskripsi jumlah barang dalam bentuk teks. Subquery digunakan untuk memilih customer yang memiliki transaksi, sedangkan stored procedure

`showDetailTransaksi` berhasil menampilkan detail transaksi dengan jumlah barang yang diubah menggunakan CASE. Implementasi ini memperkuat kemampuan MySQL dalam mengelola dan menampilkan data secara terstruktur dan informatif, serta mendukung analisis data yang lebih mendalam dalam aplikasi atau sistem yang kompleks.

**DAFTAR PUSTAKA**

Tuliskan rujukan yang anda gunakan baik website maupun buku seperti contoh dibawah.

1. Setyawati, Endang dkk. 2020. *Relational Database Management System(RDBMS)*. Banyumas: Pena Persada Redaksi.
2. Basis Data Relasional (online) [https://aws.amazon.com/id/relational-database/ *.*](https://aws.amazon.com/id/relational-database/) Diakses pada 30 April 2025.