LAPORAN TUGAS DML DAN DQL

Disusun Oleh :

Fahmi Irfan Faiz (23/520563/TK/57396)

PROGRAM STUDI SARJANA TEKNOLOGI INFORMASI

DEPARTEMEN TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS GADJAH MADA

2025

DAFTAR ISI

Contents

1. Pendahuluan

Dalam pengembangan sistem basis data, tahap manipulasi dan pengambilan data memegang peranan penting untuk menjamin sistem dapat berjalan secara dinamis dan responsif terhadap kebutuhan pengguna. Bahasa SQL *(Structured Query Language)* menyediakan berbagai perintah untuk mengelola data, di antaranya DML *(Data Manipulation Language)* dan DQL *(Data Query Language)*. DML digunakan untuk melakukan manipulasi data seperti menambahkan, mengubah, dan menghapus data pada tabel, sedangkan DQL digunakan untuk melakukan pencarian dan penarikan data dari basis data. Melalui tugas ini, dilakukan praktik langsung penerapan DML dan DQL dalam konteks sistem informasi *e-commerce* yang melibatkan entitas pengguna, admin, produk, pesanan, dan detail transaksi.

Tugas ini bertujuan untuk memahami cara menyisipkan data awal ke dalam tabel menggunakan perintah `INSERT`, menghapus data dari tabel dengan `DELETE`, serta menampilkan data menggunakan perintah `SELECT`. Dengan merancang skenario data secara realistis serta menguji berbagai *query* untuk menarik informasi dari berbagai relasi tabel, tugas ini diharapkan dapat memperkuat pemahaman konsep normalisasi data, relasi antar tabel, hingga pemanfaatan constraint seperti *foreign key* untuk menjaga integritas data.

2. Dasar Teori

1.Data Manipulation Language (DML)

DML adalah bagian dari SQL yang berfungsi untuk melakukan manipulasi data dalam basis data. Perintah-perintah DML antara lain :

- INSERT INTO digunakan untuk menambahkan data baru ke dalam tabel.
- UPDATE digunakan untuk mengubah data yang sudah ada.
- DELETE digunakan untuk menghapus data tertentu.
- TRUNCATE TABLE digunakan untuk menghapus seluruh isi tabel secara efisien tanpa menghapus strukturnya.

2. Data Query Language (DQL)

DQL digunakan untuk melakukan pencarian atau penarikan data dari tabel basis data. Perintah utama dalam DQL adalah SELECT, yang dapat dikombinasikan dengan klausa seperti WHERE, ORDER BY, GROUP BY, dan JOIN untuk memperoleh hasil query yang lebih spesifik dan informatif.

3. Relational Database

Basis data relasional menyimpan data dalam bentuk tabel-tabel yang saling terhubung melalui *primary key* dan *foreign key*. Misalnya, tabel produk terhubung dengan tabel admin melalui atribut id_admin, dan tabel pesanan terhubung dengan pengguna melalui id_pengguna. Relasi ini memungkinkan kita untuk menggabungkan data dari beberapa tabel menggunakan perintah JOIN dalam *query* SQL.

4. Constraint

Constraint adalah aturan yang diterapkan pada kolom tabel untuk menjaga validitas dan integritas data. Contohnya :

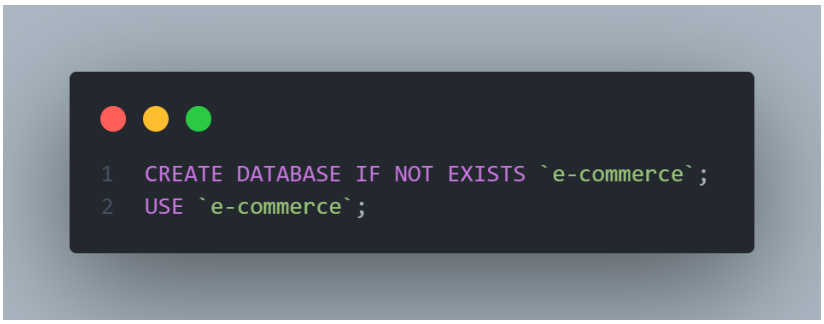
- PRIMARY KEY menjamin bahwa data di kolom tersebut bersifat unik dan tidak null.
- UNIQUE menjamin tidak ada data yang duplikat.
- FOREIGN KEY menghubungkan dua tabel dan menjaga keterkaitan data antar-tabel.
- NOT NULL memastikan kolom harus memiliki nilai.

3. Perancangan Basis Data

Perancangan Basis Data seperti *Entity Relationship Diagram (ERD)*, Relasi antar tabel, serta Desain Tabel telah dibahas secara lengkap pada Laporan Tugas Sebelumnya, sehingga pada laporan ini akan difokuskan pada proses implementasi SQL dan pembahasan hasilnya.

4. Implementasi dan Pembahasan

4.1. Membuat Database



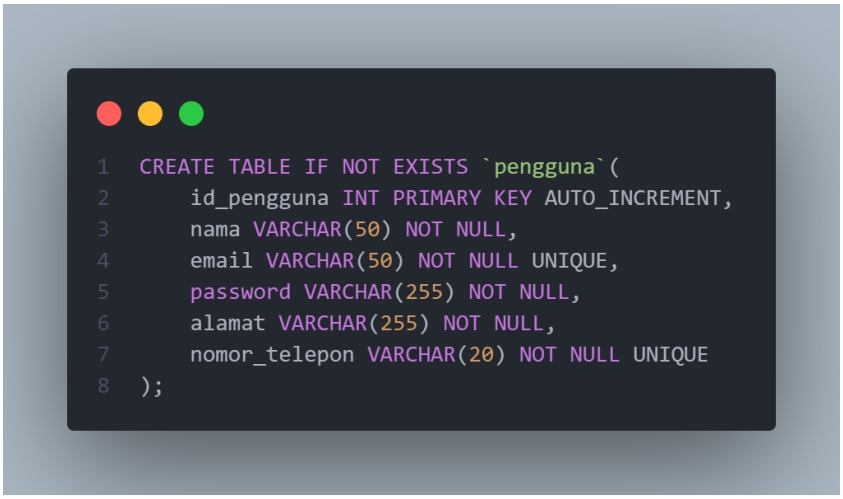
Gambar 4.1. Membuat dan menggunakan basis data e-commerce

dengan menggunakan query MySQL

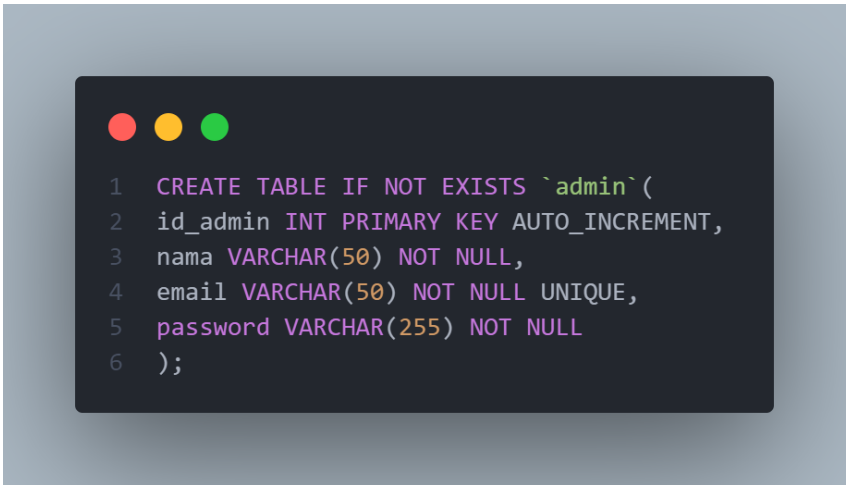
Penjelasan :

query MySQL di atas berfungsi untuk membuat basis data dengan nama "e-commerce" (jika belum ada), lalu menggunkannya sebagai basis data aktif.

4.2. Membuat Tabel Basis Data

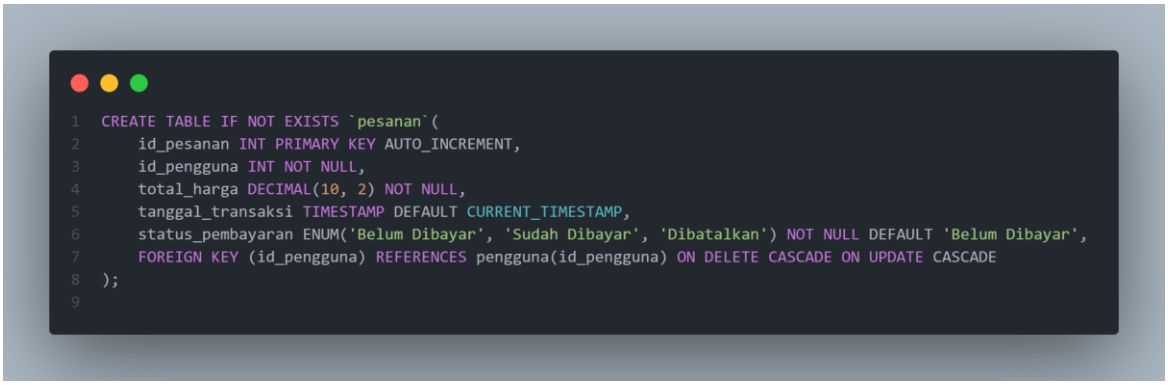


Gambar 4.2. Membuat tabel pengguna dengan menggunakan query MySQL



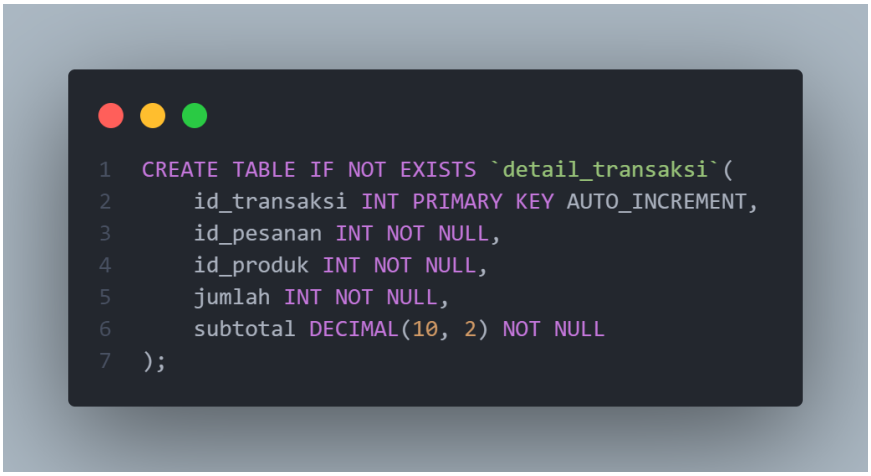
```sql
CREATE TABLE IF NOT EXISTS `admin`(
id_admin INT PRIMARY KEY AUTO_INCREMENT,
nama VARCHAR(50) NOT NULL,
email VARCHAR(50) NOT NULL UNIQUE,
password VARCHAR(255) NOT NULL
);
```

Gambar 4.3. Membuat tabel admin dengan menggunakan query MySQL



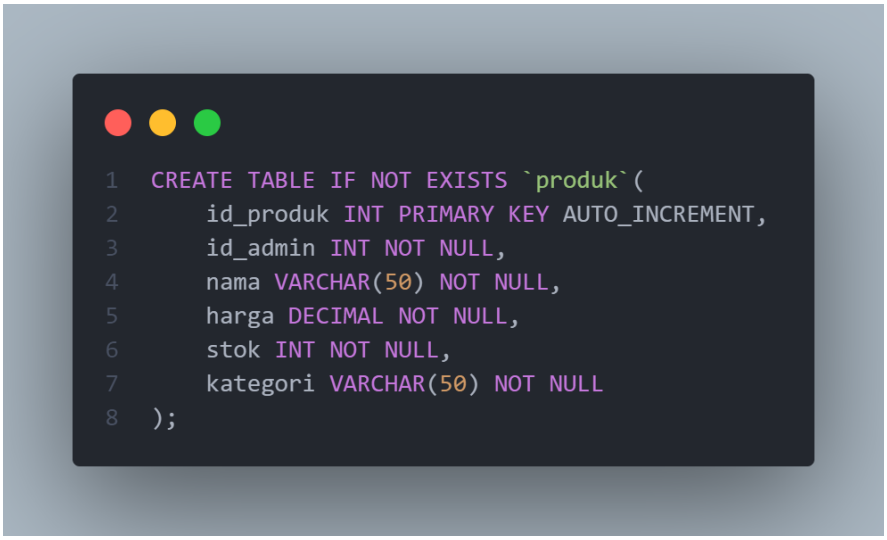
```sql
CREATE TABLE IF NOT EXISTS `pesanan`(
    id_pesanan INT PRIMARY KEY AUTO_INCREMENT,
    id_pengguna INT NOT NULL,
    total_harga DECIMAL(10, 2) NOT NULL,
    tanggal_transaksi TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status_pembayaran ENUM('Belum Dibayar', 'Sudah Dibayar', 'Dibatalkan') NOT NULL DEFAULT 'Belum Dibayar',
    FOREIGN KEY (id_pengguna) REFERENCES pengguna(id_pengguna) ON DELETE CASCADE ON UPDATE CASCADE
);

```

Gambar 4.4. Membuat tabel pesanan dengan menggunakan query MySQL.



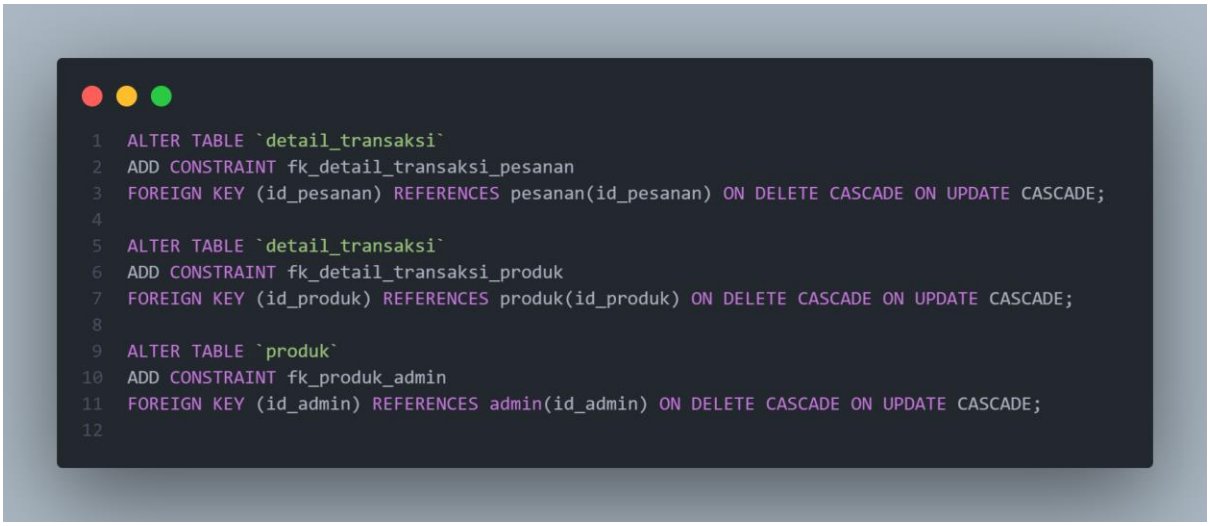
```sql
CREATE TABLE IF NOT EXISTS `detail_transaksi`(
    id_transaksi INT PRIMARY KEY AUTO_INCREMENT,
    id_pesanan INT NOT NULL,
    id_produk INT NOT NULL,
    jumlah INT NOT NULL,
    subtotal DECIMAL(10, 2) NOT NULL
);
```

Gambar 4.5. Membuat tabel detail_transaksi dengan menggunakan query MySQL



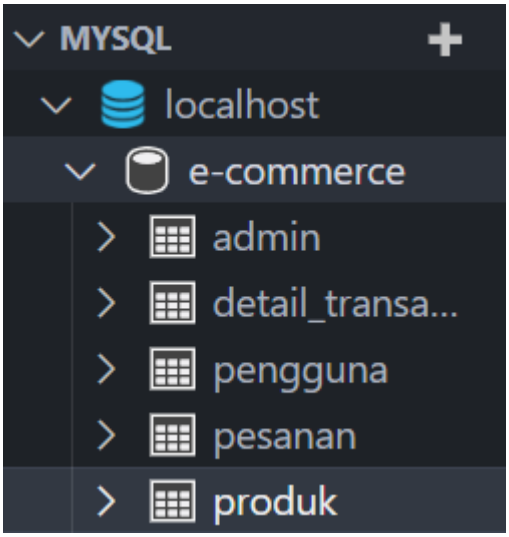
```sql
CREATE TABLE IF NOT EXISTS `produk`(
    id_produk INT PRIMARY KEY AUTO_INCREMENT,
    id_admin INT NOT NULL,
    nama VARCHAR(50) NOT NULL,
    harga DECIMAL NOT NULL,
    stok INT NOT NULL,
    kategori VARCHAR(50) NOT NULL
);
```

Gambar 4.6. Membuat tabel produk dengan menggunakan query MySQL



```sql
ALTER TABLE `detail_transaksi`
ADD CONSTRAINT fk_detail_transaksi_pesanan
FOREIGN KEY (id_pesanan) REFERENCES pesanan(id_pesanan) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `detail_transaksi`
ADD CONSTRAINT fk_detail_transaksi_produk
FOREIGN KEY (id_produk) REFERENCES produk(id_produk) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `produk`
ADD CONSTRAINT fk_produk_admin
FOREIGN KEY (id_admin) REFERENCES admin(id_admin) ON DELETE CASCADE ON UPDATE CASCADE;

```

Gambar 4.7. Menghubungkan tabel detail transaksi, produk, dan admin melalui foreign key masing-masing dengan menggunakan query MySQL



Gambar 4.8. Struktur basis data *e-commerce* dalam *localhost connection* di *software* Visual Studio Code

Penjelasan :

Berdasarkan *source code* MySQL di atas, bagian *source code* query.sql di atas adalah bagian dari proses perancangan dan implementasi struktur basis data untuk sebuah sistem *e-commerce*. *Source code* ini membuat lima buah tabel utama, yaitu pengguna, admin, pesanan, detail_transaksi, dan produk. Tabel pengguna menyimpan informasi user seperti nama, email, password, alamat, dan nomor telepon, dengan validasi unik untuk email dan nomor telepon. Tabel admin menyimpan data admin seperti nama, email, dan password, juga dengan email yang bersifat unik. Tabel pesanan mencatat transaksi yang dilakukan oleh pengguna dengan menyertakan total harga, status pembayaran yang dibatasi dengan ENUM, serta waktu transaksi yang otomatis terisi dengan waktu saat entri dilakukan. Tabel ini juga menghubungkan id_pengguna sebagai *foreign key* ke tabel pengguna.

Selanjutnya, tabel produk menyimpan daftar produk yang dikelola oleh admin, mencakup nama produk, harga, stok, dan kategori, serta referensi ke id_admin. Tabel detail_transaksi berfungsi sebagai tabel penghubung antara pesanan dan produk, untuk mencatat produk apa saja yang dibeli dalam satu pesanan, berapa jumlahnya, dan berapa subtotalnya. Tabel ini kemudian di-*alter* untuk menambahkan *foreign key* id_pesanan dan id_produk ke tabel pesanan dan produk, serta memastikan *referential integrity* melalui opsi `ON DELETE CASCADE ON UPDATE CASCADE`, yang berarti jika data dihapus atau diubah di tabel referensi, maka perubahan tersebut akan diikuti secara otomatis di tabel terkait. Selain itu, *foreign key* juga ditambahkan dari produk ke admin, yang menunjukkan bahwa setiap produk dikelola oleh seorang admin tertentu. Secara keseluruhan, potongan *source code* di atas mencerminkan perancangan basis data yang baik dengan normalisasi dan relasi antar-tabel yang konsisten.

4.3. Mengisi Tabel Basis Data

Tabel Admin

```sql
INSERT INTO admin (nama, email, password) VALUES
('Admin #1', 'admin1@example.com', 'password1'),
('Admin #2', 'admin2@example.com', 'password2'),
('Admin #3', 'admin3@example.com', 'password3'),
('Admin #4', 'admin4@example.com', 'password4'),
('Admin #5', 'admin5@example.com', 'password5');
```

Gambar 4.9. Memasukkan 5 baris *(record)* ke tabel admin dengan menggunakan perintah DML *insert* dalam MySQL

id_admin	nama	email	password
1	Admin #1	admin1@example.com	password1
2	Admin #2	admin2@example.com	password2
3	Admin #3	admin3@example.com	password3
4	Admin #4	admin4@example.com	password4
5	Admin #5	admin5@example.com	password5

Gambar 4.10. Tabel yang dihasilkan dari proses *inserting data* ke tabel admin

Tabel Pengguna

```sql
INSERT INTO pengguna (nama, email, password, alamat, nomor_telepon) VALUES
('User A', 'usera@mail.com', 'password1', 'Alamat A', '0811111111'),
('User B', 'userb@mail.com', 'password2', 'Alamat B', '0812222222'),
('User C', 'userc@mail.com', 'password3', 'Alamat C', '0813333333'),
('User D', 'userd@mail.com', 'password4', 'Alamat D', '0814444444'),
('User E', 'usere@mail.com', 'password5', 'Alamat E', '0815555555');
```

Gambar 4.11. Memasukkan 5 baris *(record)* ke tabel pengguna dengan menggunakan perintah DML *insert* dalam MySQL

id_pengguna	nama	email	password	alamat	nomor_telepon
1	User A	usera@mail.com	password1	Alamat A	0811111111
2	User B	userb@mail.com	password2	Alamat B	0812222222
3	User C	userc@mail.com	password3	Alamat C	0813333333
4	User D	userd@mail.com	password4	Alamat D	0814444444
5	User E	usere@mail.com	password5	Alamat E	0815555555

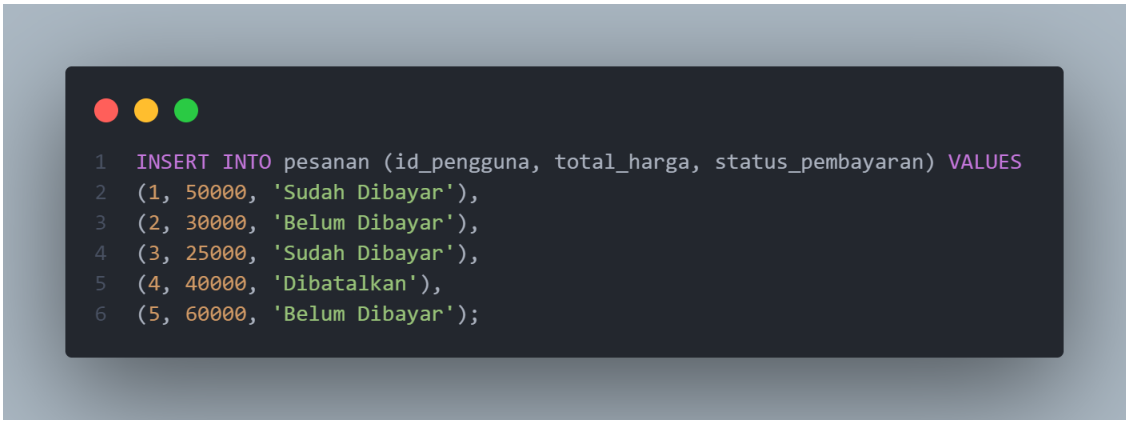
Gambar 4.12. Tabel yang dihasilkan dari proses *inserting data* ke tabel pengguna

```
INSERT INTO produk (id_admin, nama, harga, stok, kategori) VALUES
(1, 'Nasi Goreng', 10000, 50, 'Makanan'),
(2, 'Es Teh', 2000, 40, 'Minuman'),
(3, 'Burger', 15000, 30, 'Makanan'),
(4, 'Kabel USB Type-C', 25000, 20, 'Elektronik'),
(5, 'Baju Pantai', 30000, 10, 'Pakaian');
```

Gambar 4.13. Memasukkan 5 baris *(record)* ke tabel pengguna dengan menggunakan perintah DML *insert* dalam MySQL

id_produk	id_admin	nama	harga	stok	kategori
1	1	Nasi Goreng	10000	50	Makanan
2	2	Es Teh	2000	40	Minuman
3	3	Burger	15000	30	Makanan
4	4	Kabel USB Type-C	25000	20	Elektronik
5	5	Baju Pantai	30000	10	Pakaian

Gambar 4.14. Tabel yang dihasilkan dari proses *inserting data* ke tabel produk



```
INSERT INTO pesanan (id_pengguna, total_harga, status_pembayaran) VALUES
(1, 50000, 'Sudah Dibayar'),
(2, 30000, 'Belum Dibayar'),
(3, 25000, 'Sudah Dibayar'),
(4, 40000, 'Dibatalkan'),
(5, 60000, 'Belum Dibayar');
```

Gambar 4.15. Memasukkan 5 baris *(record)* ke tabel pesanan dengan menggunakan perintah DML *insert* dalam MySQL

id_pesanan	id_pengguna	total_harga	tanggal_transaksi	status_pembayaran
1	1	50000	Sat Apr 12 2025 17:21:06 GMT+0700 (Western Indonesia Time)	Sudah Dibayar
2	2	30000	Sat Apr 12 2025 17:21:06 GMT+0700 (Western Indonesia Time)	Belum Dibayar
3	3	25000	Sat Apr 12 2025 17:21:06 GMT+0700 (Western Indonesia Time)	Sudah Dibayar
4	4	40000	Sat Apr 12 2025 17:21:06 GMT+0700 (Western Indonesia Time)	Dibatalkan
5	5	60000	Sat Apr 12 2025 17:21:06 GMT+0700 (Western Indonesia Time)	Belum Dibayar

Gambar 4.16. Tabel yang dihasilkan dari proses *inserting data* ke tabel pesanan

```
INSERT INTO produk (id_admin, nama, harga, stok, kategori) VALUES
(1, 'Nasi Goreng', 10000, 50, 'Makanan'),
(2, 'Es Teh', 2000, 40, 'Minuman'),
(3, 'Burger', 15000, 30, 'Makanan'),
(4, 'Kabel USB Type-C', 25000, 20, 'Elektronik'),
(5, 'Baju Pantai', 30000, 10, 'Pakaian');
```

Gambar 4.17. Memasukkan 5 baris *(record)* ke tabel produk dengan menggunakan perintah DML *insert* dalam MySQL

id_produk	id_admin	nama	harga	stok	kategori
1	1	Nasi Goreng	10000	50	Makanan
2	2	Es Teh	2000	40	Minuman
3	3	Burger	15000	30	Makanan
4	4	Kabel USB Type-C	25000	20	Elektronik
5	5	Baju Pantai	30000	10	Pakaian

Gambar 4.18. Tabel yang dihasilkan dari proses *inserting data* ke tabel produk

Penjelasan :

source code query.sql di atas berisikan lima perintah SQL INSERT INTO dalam MySQL yang masing-masing berfungsi untuk mengisi *(seed)* data awal ke lima tabel yang berbeda dalam sebuah basis data relasional dengan detail penjelasan sebagai berikut :

1. INSERT pertama mengisi tabel admin dengan lima baris data. Kolom yang diisi adalah nama, email, dan password. Setiap baris merepresentasikan akun admin berbeda, seperti "Admin #1" hingga "Admin #5", dengan email dan password yang unik untuk masing-masing. Ini memungkinkan sistem memiliki beberapa admin

terdaftar di awal, yang mungkin bertugas mengelola produk, pengguna, atau pesanan.

2. INSERT kedua mengisi tabel pengguna (user) dengan lima entri. Selain nama, email, dan password seperti pada tabel admin, tabel pengguna ini juga menyimpan alamat dan nomor telepon. Kolom-kolom ini umum digunakan dalam sistem *e-commerce* atau layanan untuk identifikasi dan pengiriman. Data ini mendemonstrasikan pendaftaran pengguna biasa dalam sistem.

3. INSERT ketiga memasukkan data ke dalam tabel produk. Setiap produk berisi id_admin (yang menunjukkan admin mana yang menambahkan produk tersebut), nama produk, harga, stok, dan kategori. Contohnya, produk "Nasi Goreng" dibuat oleh admin dengan id 1, memiliki harga Rp10.000, stok 50, dan dikategorikan sebagai “Makanan”. Ini menunjukkan adanya relasi antara produk dan admin sebagai pemilik atau pengelola produk.

4. INSERT keempat mengisi tabel pesanan *(order)* dengan lima transaksi. Tiap pesanan berisi id_pengguna (pemilik pesanan), total_harga pesanan, dan status_pembayaran. Status bisa berupa “Sudah Dibayar”, “Belum Dibayar”, atau “Dibatalkan”. Ini mencerminkan kondisi transaksi dalam sistem, baik yang sudah selesai, tertunda, atau batal.

5. INSERT terakhir mengisi tabel detail_transaksi, yang mencatat rincian dari masing-masing pesanan. Kolom yang diisi meliputi id_pesanan, id_produk, jumlah produk yang dipesan, dan subtotal harga (hasil dari harga produk dikali jumlah). Ini penting untuk menunjukkan isi dari tiap pesanan. Misalnya, entri pertama menunjukkan bahwa pesanan ke-1 berisi 2 unit produk dengan id 1 (yaitu "Nasi Goreng"), totalnya Rp20.000.

Secara keseluruhan, bagian *source code* query.sql ini dirancang untuk mengisi database dengan *dummy data* awal yang mencerminkan struktur relasional antar entitas dalam sebuah sistem manajemen penjualan seperti *e-commerce* atau sistem pemesanan makanan. Relasi antar tabel juga sudah tergambarkan dengan penggunaan *foreign key* secara implisit, seperti id_admin di tabel produk, id_pengguna di tabel pesanan, dan id_pesanan serta id_produk di tabel detail_transaksi.

4.4. Mengubah 1 Data Pada Tabel Basis Data

```sql
UPDATE admin SET nama = 'Admin Satu' WHERE id_admin = 1;
UPDATE pengguna SET alamat = 'Alamat Baru A' WHERE id_pengguna = 1;
UPDATE produk SET harga = 11000 WHERE id_produk = 1;
UPDATE pesanan SET status_pembayaran = 'Sudah Dibayar' WHERE id_pesanan = 2;
UPDATE detail_transaksi SET jumlah = 3, subtotal = 30000 WHERE id_transaksi = 1;
```

Gambar 4.19. Mengubah 1 baris data pada masing-masing tabel dengan menggunakan perintah DML SQL UPDATE

id_admin	nama	email	password
1	Admin Satu	admin1@example.com	password1
2	Admin #2	admin2@example.com	password2
3	Admin #3	admin3@example.com	password3
4	Admin #4	admin4@example.com	password4
5	Admin #5	admin5@example.com	password5

Gambar 4.20. Tabel Admin setelah baris nama dengan

id_admin = 1 di-*update* menggunakan perintah SQL DML UPDATE

id_transaksi	id_pesanan	id_produk	jumlah	subtotal
1	1	1	3	30000
2	2	2	1	20000
3	3	3	1	15000
4	4	4	1	25000
5	5	5	2	60000

Gambar 4.21. Tabel Detail Transaksi setelah baris jumlah dengan id_transaksi = 1

di-*update* menggunakan perintah SQL DML UPDATE

id_pengguna	nama	email	password	alamat	nomor_telepon
1	User A	usera@mail.com	password1	Alamat Baru A	0811111111
2	User B	userb@mail.com	password2	Alamat B	0812222222
3	User C	userc@mail.com	password3	Alamat C	0813333333
4	User D	userd@mail.com	password4	Alamat D	0814444444
5	User E	usere@mail.com	password5	Alamat E	0815555555

Gambar 4.22. Tabel Pengguna setelah baris Alamat dengan id_pengguna = 1

di-*update* menggunakan perintah SQL DML UPDATE

id_pesanan	id_pengguna	total_harga	tanggal_transaksi	status_pembayaran
1	1	50000	Sat Apr 12 2025 17:21:06 GMT+0700 (Western Indonesia Time)	Sudah Dibayar
2	2	30000	Sat Apr 12 2025 17:21:06 GMT+0700 (Western Indonesia Time)	Sudah Dibayar
3	3	25000	Sat Apr 12 2025 17:21:06 GMT+0700 (Western Indonesia Time)	Sudah Dibayar
4	4	40000	Sat Apr 12 2025 17:21:06 GMT+0700 (Western Indonesia Time)	Dibatalkan
5	5	60000	Sat Apr 12 2025 17:21:06 GMT+0700 (Western Indonesia Time)	Belum Dibayar

Gambar 4.23. Tabel Pesanan setelah baris status_pembayaran dengan id_pesanan = 2 di-*update* menggunakan perintah SQL DQL UPDATE

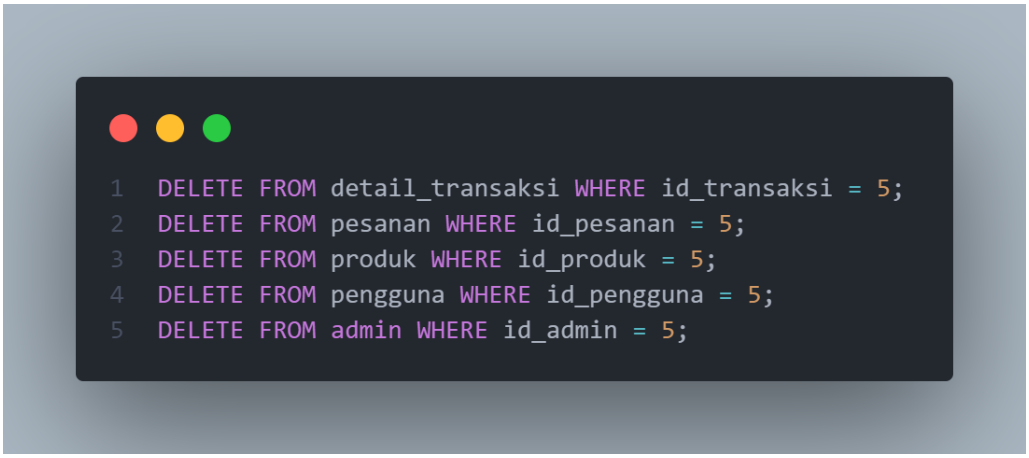
id_produk	id_admin	nama	harga	stok	kategori
1	1	Nasi Goreng	11000	50	Makanan
2	2	Es Teh	2000	40	Minuman
3	3	Burger	15000	30	Makanan
4	4	Kabel USB Type-C	25000	20	Elektronik
5	5	Baju Pantai	30000	10	Pakaian

Gambar 4.24. Tabel Produk setelah baris harga dengan id_produk = 1 di-*update* menggunakan perintah SQL DQL UPDATE

Penjelasan :

Seperti yang nampak pada lampiran masing-masing tabel, perintah SQL DQL UPDATE berfungsi untuk mengubah 1 baris data pada suatu kolom tertentu dengan *selection* WHERE $< condition >$ dan perintah DQL UPDATE tersebut juga berhasil memperbarui 1 baris data pada setiap tabel dengan variasi perintah UPDATE untuk setiap tabel.

4.5. Menghapus 1 Data Pada Tabel Basis Data



Gambar 4.25. Menghapus 1 baris data dari setiap tabel dengan perintah SQL DML DELETE

id_admin	nama	email	password
1	Admin Satu	admin1@example.com	password1
2	Admin #2	admin2@example.com	password2
3	Admin #3	admin3@example.com	password3
4	Admin #4	admin4@example.com	password4

Gambar 4.26. Tabel Admin setelah baris dengan id_admin = 5 dihapus dengan perintah SQL DML DELETE

id_produk	id_admin	nama	harga	stok	kategori
1	1	Nasi Goreng	11000	50	Makanan
2	2	Es Teh	2000	40	Minuman
3	3	Burger	15000	30	Makanan
4	4	Kabel USB Type-C	25000	20	Elektronik

Gambar 4.27. Tabel Produk setelah baris dengan id_produk = 5 dihapus dengan perintah SQL DML DELETE

id_transaksi	id_pesanan	id_produk	jumlah	subtotal
1	1	1	3	30000
2	2	2	1	20000
3	3	3	1	15000
4	4	4	1	25000

Gambar 4.28. Tabel Detail Transaksi setelah baris dengan id_transaksi = 5 dihapus dengan perintah SQL DML DELETE

id_pengguna	nama	email	password	alamat	nomor_telepon
1	User A	usera@mail.com	password1	Alamat Baru A	0811111111
2	User B	userb@mail.com	password2	Alamat B	0812222222
3	User C	userc@mail.com	password3	Alamat C	0813333333
4	User D	userd@mail.com	password4	Alamat D	0814444444

Gambar 4.29. Tabel Pengguna setelah baris dengan id_pengguna = 5 dihapus dengan perintah SQL DML DELETE

id_pesanan	id_pengguna	total_harga	tanggal_transaksi	status_pembayaran
1	1	50000	Sat Apr 12 2025 21:20:19 GMT+0700 (Western Indonesia Time)	Sudah Dibayar
2	2	30000	Sat Apr 12 2025 21:20:19 GMT+0700 (Western Indonesia Time)	Sudah Dibayar
3	3	25000	Sat Apr 12 2025 21:20:19 GMT+0700 (Western Indonesia Time)	Sudah Dibayar
4	4	40000	Sat Apr 12 2025 21:20:19 GMT+0700 (Western Indonesia Time)	Dibatalkan

Gambar 4.30. Tabel Pesanan setelah baris dengan id_pesanan = 5 dihapus dengan perintah SQL DML DELETE

Penjelasan :

Berdasarkan potongan *source code* query.sql di atas, semua baris kelima pada setiap pada tabel di atas dengan *identifier* tertentu berhasil dihapus dari tabel dengan menerapkan perintah SQL DML DELETE

4.6. Menerapkan 3 Query Tertentu Pada Setiap Tabel

```
5. -- Table Pengguna
6. SELECT alamat, COUNT(*) FROM pengguna GROUP BY alamat;
7. SELECT alamat, COUNT(*) FROM pengguna GROUP BY alamat HAVING COUNT(*) >
   0;
8. SELECT * FROM pengguna ORDER BY nama ASC;
9.
10.-- Table Admin
11.SELECT COUNT(*) AS total_admin FROM admin;
12.SELECT * FROM admin WHERE email LIKE '%admin%';
13.SELECT nama FROM admin UNION SELECT nama FROM pengguna;
14.
15.-- Table Produk
16.SELECT kategori, AVG(harga) FROM produk GROUP BY kategori;
17.SELECT kategori, AVG(harga) FROM produk GROUP BY kategori HAVING
   AVG(harga) > 10000;
18.SELECT * FROM produk ORDER BY stok DESC;
19.
20.-- Table Pesanan
21.SELECT status_pembayaran, COUNT(*) FROM pesanan GROUP BY
   status_pembayaran;
22.SELECT * FROM pesanan ORDER BY total_harga DESC;
23.SELECT * FROM pesanan WHERE total_harga > 30000;
24.
25.-- Table Detail Transaksi
26.SELECT id_produk, SUM(subtotal) AS total FROM detail_transaksi GROUP BY
   id_produk;
27.SELECT * FROM detail_transaksi WHERE jumlah > 1;
28.SELECT * FROM detail_transaksi ORDER BY subtotal DESC;
```

alamat	COUNT(*)
Alamat Baru A	1
Alamat B	1
Alamat C	1
Alamat D	1

alamat	COUNT(*)
Alamat Baru A	1
Alamat B	1
Alamat C	1
Alamat D	1

id_pengguna	nama	email	password	alamat	nomor_telepon
1	User A	usera@mail.com	password1	Alamat Baru A	0811111111
2	User B	userb@mail.com	password2	Alamat B	0812222222
3	User C	userc@mail.com	password3	Alamat C	0813333333
4	User D	userd@mail.com	password4	Alamat D	0814444444

Gambar 4.31. Menerapkan 3 query khusus pada Tabel Pengguna

total_admin
4

id_admin	nama	email	password
1	Admin Satu	admin1@example.com	password1
2	Admin #2	admin2@example.com	password2
3	Admin #3	admin3@example.com	password3
4	Admin #4	admin4@example.com	password4

nama
Admin Satu
Admin #2
Admin #3
Admin #4
User A
User B
User C
User D

Gambar 4.32. Menerapkan 3 query khusus pada Tabel Admin

kategori	AVG(harga)
Makanan	13000
Minuman	2000
Elektronik	25000

kategori	AVG(harga)
Makanan	13000
Elektronik	25000

id_produk	id_admin	nama	harga	stok	kategori
1	1	Nasi Goreng	11000	50	Makanan
2	2	Es Teh	2000	40	Minuman
3	3	Burger	15000	30	Makanan
4	4	Kabel USB Type-C	25000	20	Elektronik

Gambar 4.33. Menerapkan 3 query khusus pada Tabel Produk

status_pembayaran	COUNT(*)
Sudah Dibayar	3
Dibatalkan	1

id_pesanan	id_pengguna	total_harga	tanggal_transaksi	status_pembayaran
1	1	50000	Sat Apr 12 2025 21:20:19 GMT+0700 (Western Indonesia Time)	Sudah Dibayar
4	4	40000	Sat Apr 12 2025 21:20:19 GMT+0700 (Western Indonesia Time)	Dibatalkan
2	2	30000	Sat Apr 12 2025 21:20:19 GMT+0700 (Western Indonesia Time)	Sudah Dibayar
3	3	25000	Sat Apr 12 2025 21:20:19 GMT+0700 (Western Indonesia Time)	Sudah Dibayar

id_pesanan	id_pengguna	total_harga	tanggal_transaksi	status_pembayaran
1	1	50000	Sat Apr 12 2025 21:20:19 GMT+0700 (Western Indonesia Time)	Sudah Dibayar
4	4	40000	Sat Apr 12 2025 21:20:19 GMT+0700 (Western Indonesia Time)	Dibatalkan

Gambar 4.34 Menerapkan 3 query khusus pada Tabel Pesanan

id_produk	total
1	30000
2	20000
3	15000
4	25000

id_transaksi	id_pesanan	id_produk	jumlah	subtotal
1	1	1	3	30000

id_transaksi	id_pesanan	id_produk	jumlah	subtotal
1	1	1	3	30000
4	4	4	1	25000
2	2	2	1	20000
3	3	3	1	15000

Gambar 4.35. Menerapkan 3 query khusus pada Tabel Detail Transaksi

4.7. Menerapkan 2 Operasi JOIN dan SUBQUERY dari Beberapa Tabel

```sql
-- Join: pengguna dengan pesanan
SELECT p.nama, ps.total_harga, ps.status_pembayaran
FROM pengguna p
JOIN pesanan ps ON p.id_pengguna = ps.id_pengguna;

-- Join: detail_transaksi dengan produk dan pesanan
SELECT dt.id_transaksi, pr.nama AS nama_produk, dt.jumlah, dt.subtotal, ps.tanggal_transaksi
FROM detail_transaksi dt
JOIN produk pr ON dt.id_produk = pr.id_produk
JOIN pesanan ps ON dt.id_pesanan = ps.id_pesanan;

-- Subquery: produk dengan harga di atas rata-rata
SELECT * FROM produk
WHERE harga > (SELECT AVG(harga) FROM produk);

-- Subquery: pengguna yang memiliki pesanan
SELECT * FROM pengguna
WHERE id_pengguna IN (SELECT id_pengguna FROM pesanan);
```

nama	total_harga	status_pembayaran
User A	50000	Sudah Dibayar
User B	30000	Sudah Dibayar
User C	25000	Sudah Dibayar
User D	40000	Dibatalkan

Gambar 4.35. Operasi JOIN Antara Tabel Pengguna dengan Tabel Pesanan

id_transaksi	nama_produk	jumlah	subtotal	tanggal_transaksi
1	Nasi Goreng	3	30000	Sat Apr 12 2025 21:20:19 GMT+0700 (Western Indonesia Time)
2	Es Teh	1	20000	Sat Apr 12 2025 21:20:19 GMT+0700 (Western Indonesia Time)
3	Burger	1	15000	Sat Apr 12 2025 21:20:19 GMT+0700 (Western Indonesia Time)
4	Kabel USB Type-C	1	25000	Sat Apr 12 2025 21:20:19 GMT+0700 (Western Indonesia Time)

Gambar 4.36. Operasi JOIN Antara Tabel Detail Transaksi, Tabel Produk, dan Pesanan

id_produk	id_admin	nama	harga	stok	kategori
3	3	Burger	15000	30	Makanan
4	4	Kabel USB Type-C	25000	20	Elektronik

Gambar 4.37. Operasi SUBQUERY Produk dengan Harga Di Atas Rata-Rata

id_pengguna	nama	email	password	alamat	nomor_telepon
1	User A	usera@mail.com	password1	Alamat Baru A	0811111111
2	User B	userb@mail.com	password2	Alamat B	0812222222
3	User C	userc@mail.com	password3	Alamat C	0813333333
4	User D	userd@mail.com	password4	Alamat D	0814444444

Gambar 4.38. Operasi SUBQUERY pengguna yang memiliki pesanan

5. Kesimpulan dan Saran

5.1.Kesimpulan

Dari kegiatan pembuatan dan manipulasi database e-commerce menggunakan perintah DML dan DQL, dapat disimpulkan bahwa:

1. **Perintah DML** seperti INSERT, UPDATE, dan DELETE memungkinkan kita untuk mengelola data dalam tabel secara efisien, baik untuk menambahkan data baru, memperbarui data yang sudah ada, maupun menghapus data yang tidak diperlukan.
2. **Perintah DQL**, khususnya SELECT, sangat penting untuk menampilkan dan menganalisis data yang tersimpan di dalam database, terutama jika dikombinasikan dengan klausa seperti GROUP BY, ORDER BY, HAVING, dan UNION.

3. Struktur database yang dirancang dengan hubungan antar tabel (melalui foreign key) memberikan fleksibilitas dan konsistensi data, serta mendukung integritas referensial.

4. Melalui implementasi langsung, mahasiswa menjadi lebih memahami bagaimana konsep-konsep SQL diterapkan dalam konteks dunia nyata, khususnya pada sistem informasi e-commerce.

5.2. Saran

Agar pengelolaan basis data dapat dilakukan secara lebih optimal di masa depan, maka disarankan :

1. **Validasi dan enkripsi data** penting seperti password sebaiknya diterapkan, misalnya menggunakan fungsi hash seperti SHA2() agar keamanan data pengguna lebih terjamin.

2. **Penggunaan stored procedure dan trigger** dapat dipertimbangkan untuk mengotomatisasi proses tertentu dan menjaga konsistensi data tanpa harus selalu melakukan query manual.

3. Perlu **pengujian data** secara berkala untuk memastikan tidak ada anomali atau inkonsistensi akibat operasi DML.

4. **Optimisasi query SELECT**, terutama untuk tabel yang memiliki banyak data, dapat dilakukan dengan indexing dan analisis query plan agar performa database tetap terjaga.

5. Terus mengembangkan struktur database seiring kompleksitas sistem bertambah, dengan memperhatikan prinsip-prinsip normalisasi dan efisiensi relasi antar tabel.

6. Lampiran

6.1. Lampiran *Source Code* MySQL

```
29.-- 2. Membuat Database dan Tabel e-commerce
30.
31.-- CREATE DATABASE IF NOT EXISTS `e-commerce`;
32.USE `e-commerce`;
33.
34.CREATE TABLE IF NOT EXISTS `pengguna`(
```

```
    id_pengguna INT PRIMARY KEY AUTO_INCREMENT,
    nama VARCHAR(50) NOT NULL,
    email VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    alamat VARCHAR(255) NOT NULL,
    nomor_telepon VARCHAR(20) NOT NULL UNIQUE
);
CREATE TABLE IF NOT EXISTS `admin`(
id_admin INT PRIMARY KEY AUTO_INCREMENT,
nama VARCHAR(50) NOT NULL,
email VARCHAR(50) NOT NULL UNIQUE,
password VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS `pesanan`(
    id_pesanan INT PRIMARY KEY AUTO_INCREMENT,
    id_pengguna INT NOT NULL,
    total_harga DECIMAL(10, 2) NOT NULL,
    tanggal_transaksi TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status_pembayaran ENUM('Belum Dibayar', 'Sudah Dibayar',
'Dibatalkan') NOT NULL DEFAULT 'Belum Dibayar',
    FOREIGN KEY (id_pengguna) REFERENCES pengguna(id_pengguna) ON
DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS `detail_transaksi`(
    id_transaksi INT PRIMARY KEY AUTO_INCREMENT,
    id_pesanan INT NOT NULL,
    id_produk INT NOT NULL,
    jumlah INT NOT NULL,
    subtotal DECIMAL(10, 2) NOT NULL
);

CREATE TABLE IF NOT EXISTS `produk`(
    id_produk INT PRIMARY KEY AUTO_INCREMENT,
    id_admin INT NOT NULL,
    nama VARCHAR(50) NOT NULL,
    harga DECIMAL NOT NULL,
    stok INT NOT NULL,
    kategori VARCHAR(50) NOT NULL
);

ALTER TABLE `detail_transaksi`
ADD CONSTRAINT fk_detail_transaksi_pesanan
FOREIGN KEY (id_pesanan) REFERENCES pesanan(id_pesanan) ON DELETE
CASCADE ON UPDATE CASCADE;

ALTER TABLE `detail_transaksi`
```

```
ADD CONSTRAINT fk_detail_transaksi_produk
FOREIGN KEY (id_produk) REFERENCES produk(id_produk) ON DELETE CASCADE
ON UPDATE CASCADE;

ALTER TABLE `produk`
ADD CONSTRAINT fk_produk_admin
FOREIGN KEY (id_admin) REFERENCES admin(id_admin) ON DELETE CASCADE ON
UPDATE CASCADE;

3. Mengisi Tabel Dengan Data Sebanyak 5 Baris (Record)

INSERT INTO admin (nama, email, password) VALUES
('Admin #1', 'admin1@example.com', 'password1'),
('Admin #2', 'admin2@example.com', 'password2'),
('Admin #3', 'admin3@example.com', 'password3'),
('Admin #4', 'admin4@example.com', 'password4'),
('Admin #5', 'admin5@example.com', 'password5');

INSERT INTO pengguna (nama, email, password, alamat, nomor_telepon)
VALUES
('User A', 'usera@mail.com', 'password1', 'Alamat A', '0811111111'),
('User B', 'userb@mail.com', 'password2', 'Alamat B', '0812222222'),
('User C', 'userc@mail.com', 'password3', 'Alamat C', '0813333333'),
    ('User D', 'userd@mail.com', 'password4', 'Alamat D',
'0814444444'),
    ('User E', 'usere@mail.com', 'password5', 'Alamat E',
'0815555555');

    INSERT INTO produk (id_admin, nama, harga, stok, kategori) VALUES
    (1, 'Nasi Goreng', 10000, 50, 'Makanan'),
    (2, 'Es Teh', 2000, 40, 'Minuman'),
    (3, 'Burger', 15000, 30, 'Makanan'),
    (4, 'Kabel USB Type-C', 25000, 20, 'Elektronik'),
    (5, 'Baju Pantai', 30000, 10, 'Pakaian');

    INSERT INTO pesanan (id_pengguna, total_harga, status_pembayaran)
VALUES
    (1, 50000, 'Sudah Dibayar'),
    (2, 30000, 'Belum Dibayar'),
    (3, 25000, 'Sudah Dibayar'),
    (4, 40000, 'Dibatalkan'),
    (5, 60000, 'Belum Dibayar');

    INSERT INTO detail_transaksi (id_pesanan, id_produk, jumlah,
subtotal) VALUES
    (1, 1, 2, 20000),
    (2, 2, 1, 20000),
    (3, 3, 1, 15000),
```

```
121.      (4, 4, 1, 25000),
122.      (5, 5, 2, 60000);
123.
124.      4. Mengubah 1 Data Pada Setiap Tabel
125.      UPDATE admin SET nama = 'Admin Satu' WHERE id_admin = 1;
126.      UPDATE pengguna SET alamat = 'Alamat Baru A' WHERE id_pengguna =
   1;
127.      UPDATE produk SET harga = 11000 WHERE id_produk = 1;
128.      UPDATE pesanan SET status_pembayaran = 'Sudah Dibayar' WHERE
   id_pesanan = 2;
129.      UPDATE detail_transaksi SET jumlah = 3, subtotal = 30000 WHERE
   id_transaksi = 1;
130.
131.      5. Menghapus 1 Data Pada Setiap Tabel
132.      DELETE FROM detail_transaksi WHERE id_transaksi = 5;
133.      DELETE FROM pesanan WHERE id_pesanan = 5;
134.      DELETE FROM produk WHERE id_produk = 5;
135.      DELETE FROM pengguna WHERE id_pengguna = 5;
136.      DELETE FROM admin WHERE id_admin = 5;
137.
138.      Melakukan 3 Operasi Query Pada Setiap Tabel Dengan Perintah
   SELECT Tertentu
139.      Table Pengguna
140.      SELECT alamat, COUNT(*) FROM pengguna GROUP BY alamat;
141.      SELECT alamat, COUNT(*) FROM pengguna GROUP BY alamat HAVING
   COUNT(*) > 0;
142.      SELECT * FROM pengguna ORDER BY nama ASC;
143.
144.      Table Admin
145.      SELECT COUNT(*) AS total_admin FROM `admin`;
146.      SELECT * FROM `admin` WHERE email LIKE '%admin%';
147.      SELECT nama FROM `admin` UNION SELECT nama FROM pengguna;
148.
149.      -- Table Produk
150.      SELECT kategori, AVG(harga) FROM produk GROUP BY kategori;
151.      SELECT kategori, AVG(harga) FROM produk GROUP BY kategori HAVING
   AVG(harga) > 10000;
152.      SELECT * FROM produk ORDER BY stok DESC;
153.
154.      -- Table Pesanan
155.      SELECT status_pembayaran, COUNT(*) FROM pesanan GROUP BY
   status_pembayaran;
156.      SELECT * FROM pesanan ORDER BY total_harga DESC;
157.      SELECT * FROM pesanan WHERE total_harga > 30000;
158.
159.      -- Table Detail Transaksi
160.      SELECT id_produk, SUM(subtotal) AS total FROM detail_transaksi
   GROUP BY id_produk;
```

```sql
SELECT * FROM detail_transaksi WHERE jumlah > 1;
SELECT * FROM detail_transaksi ORDER BY subtotal DESC;

-- 7. Join dan subquery

-- Join: pengguna dengan pesanan
SELECT p.nama, ps.total_harga, ps.status_pembayaran
FROM pengguna p
JOIN pesanan ps ON p.id_pengguna = ps.id_pengguna;

-- Join: detail_transaksi dengan produk dan pesanan
SELECT dt.id_transaksi, pr.nama AS nama_produk, dt.jumlah,
dt.subtotal, ps.tanggal_transaksi
FROM detail_transaksi dt
JOIN produk pr ON dt.id_produk = pr.id_produk
JOIN pesanan ps ON dt.id_pesanan = ps.id_pesanan;

-- Subquery: produk dengan harga di atas rata-rata
SELECT * FROM produk
WHERE harga > (SELECT AVG(harga) FROM produk);

-- Subquery: pengguna yang memiliki pesanan
SELECT * FROM pengguna
WHERE id_pengguna IN (SELECT id_pengguna FROM pesanan);

```

6.2. Lampiran *Source Code* MySQL (GitHub)

https://github.com/fahmiirfanfaiz/dml-and-dql-assignment-repository