

**LAPORAN TUGAS BESAR
PROYEK BASIS DATA LANJUT**



DISUSUN OLEH :

- | | |
|---------------------------|-------------|
| 1. Hesi Desta Lestari | (G1A023006) |
| 2. Diosi Putri Arlita | (G1A023012) |
| 3. Intan Oktaviani Presia | (G1A023014) |
| 4. Fassrah Putra Gunawan | (G1A023038) |

Nama Asisten Dosen :

- | | |
|--------------------------------|-------------|
| 1. Merly Yuni Purnama | (G1A022006) |
| 2. Reksi Hendra Pratama | (G1A022032) |
| 3. Sinta Ezra Wati Gulo | (G1A022040) |
| 4. Fadlan dwi Febrio | (G1A022051) |
| 5. Torang Four Yones Manullang | (G1A022052) |
| 6. Wahyu Ozorah Manurung | (G1A022060) |
| 7. Shalaudin Muhammad Sah | (G1A022070) |
| 8. Dian Ardiyanti Saputri | (G1A022084) |

Dosen Pengampu:

1. Dr., Endina Putri Puwandari, S.T., M.Kom
2. Tiara Eka Putri, S.T., M.Kom.

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS BENGKULU

KATA PENGANTAR

Puji syukur kehadiran Tuhan yang Maha Kuasa atas karunia-Nya, kami diberikan kesehatan jasmani dan rohani. Sehingga kami dapat menyelesaikan laporan ini dengan tepat waktu tanpa adanya halangan. Selesaiannya laporan ini dengan baik, tentunya tak lepas dari peran dan bantuan dari beberapa pihak.

Laporan ini disusun dalam rangka memenuhi tugas kuliah Proyek Basis Data Lanjut. Laporan ini merupakan bukti komitmen kami dalam memperoleh pemahaman yang mendalam tentang konsep-konsep yang diajarkan dalam mata kuliah Proyek Basis Data Lanjut. Tugas besar praktikum ini tidak hanya menjadi suatu kewajiban akademis semata, tetapi juga merupakan kesempatan bagi kami untuk mengaplikasikan pengetahuan yang kami peroleh selama mengikuti praktikum.

Kami berharap bahwa laporan ini dapat memberikan kontribusi positif dalam pemahaman akan materi praktikum yang telah kami pelajari. Kami ingin menyampaikan terima kasih kepada semua pihak yang telah membantu kami dalam menyelesaikan tugas ini, baik itu dosen pengampu, asisten praktikum, maupun teman-teman sejawat yang turut berkontribusi dalam proses pembelajaran kami. Akhir kata, kami berharap laporan ini dapat bermanfaat bagi pembaca dan menjadi salah satu sumbangan kecil kami dalam pengembangan ilmu pengetahuan.

Bengkulu, 5 Mei 2025

Kelompok 5

DAFTAR ISI

KATA PENGANTAR.....	i
DAFTAR ISI	ii
BAB I LANDASAN TEORI.....	1
A. Database Management System	1
B. MySQL.....	1
C. Pengertian DDL	3
D. Pengertian DCL.....	3
BAB II SOAL DAN PEMBAHASAN	4
BAB III KESIMPULAN	25
A. Kesimpulan.....	25
B. Saran	25
DAFTAR PUSTAKA.....	26
LAMPIRAN.....	27
A. Jobdesk.....	27
B. Dokumentasi.....	27

BAB 1

LANDASAN TEORI

A. Database Management System

Database Management System (DBMS) merupakan perangkat lunak untuk mengendalikan pembuatan, pemeliharaan, pengolahan, dan penggunaan data yang berskala besar. Beberapa DBMS yang digunakan adalah MySQL dan MariaDB. Berdasarkan survey yang dilakukan, MySQL dan MariaDB merupakan DBMS yang banyak digunakan sebagai contoh survey yang terdapat pada db-engines.com DB-Engines Ranking menempatkan MySQL pada posisi ke-2 sedangkan MariaDB pada posisi ke-20 namun pada survey yang terdapat di serverwatch.com Top 10 Enterprise Database System Of 2016, MariaDB menempati posisi ke-6 dan MySQL menempati posisi ke-7.

DBMS juga dirancang untuk memudahkan memanipulasi data. DBMS sudah menjadi peran atau kunci utama serta bagian standar di bagian pendukung sebuah perusahaan. Adapun bahasa dalam *Database Management System* (DBMS) sebagai berikut :

- 1.Data Definition Language (DDL)
- 2.Data Manipulation Language (DML)
- 3.Data Control Language (DCL)

Query adalah semacam kemampuan untuk menampilkan suatu data dari database dimana mengambil dari tabel-tabel yang ada dalam database, namun tabel tersebut tidak semua ditampilkan sesuai yang kita inginkan. Data apa yang ingin kita tampilkan. Misalnya, data pinjaman dengan buku pinjaman, maka nanti akan mengambil data dari tabel peminjam dan tabel buku.

B. MySQL

MYSQL menurut Raharjo (2011), merupakan RDBMS (server database) yang mengelola database dengan cepat menampung dalam jumlah sangat besar dan dapat di akses oleh banyak user. Sedangkan menurut Kadir (2008) MySQL adalah sebuah software open source yang digunakan untuk membuat sebuah database. Berdasarkan pendapat yang dikemukakan di atas dapat ditarik

kesimpulan bahwa MySQL adalah suatu software atau program yang digunakan untuk membuat sebuah basis data yang bersifat open source.

MariaDB merupakan salah satu database server yang digunakan untuk menyimpan dan manajemen data. MariaDB tidak jauh berbeda dengan MySQL, karena MariaDB merupakan versi pengembangan terbuka dan mandiri dari MySQL. Sejak diakuisisinya MySQL oleh Oracle pada September 2010, Monty Program sebagai penulis awal kode sumber MySQL memisahkan diri dari pengembangan dan membuat versi yang lebih mandiri yakni MariaDB. Sampai saat ini, sudah banyak yang telah melakukan migrasi dari MySQL ke MariaDB, contohnya perusahaan raksasa Google dan juga situs besar seperti Wikipedia. Salah satu kelebihan MariaDB adalah karena performannya yang cukup bagus dan tidak berat serta kompatibel dengan MySQL. MariaDB juga kompatibel dengan berbagai macam platform seperti LINUX, Windows, MacOS, FreeBSD, Solaris. MySQL dikembangkan pertama kali oleh perusahaan Swedia, MySQL AB, dan kini dimiliki serta dikembangkan oleh Oracle Corporation. MySQL banyak digunakan dalam pengembangan aplikasi web karena bersifat cepat, andal, dan dapat dijalankan di berbagai sistem operasi seperti Windows, Linux, dan macOS. MySQL berfungsi untuk menyimpan, mengatur, dan mengelola data dalam bentuk tabel yang saling terhubung satu sama lain melalui relasi. MySQL juga mendukung berbagai fitur seperti transaksi, replikasi data, dan sistem keamanan yang memungkinkan pengaturan hak akses pengguna. Selain itu, MySQL mudah diintegrasikan dengan berbagai bahasa pemrograman seperti PHP, Python, dan Java, sehingga sangat populer digunakan dalam pembuatan situs web dinamis dan aplikasi berbasis database. Karena kemudahan penggunaan dan performanya yang baik, MySQL menjadi salah satu pilihan utama dalam pengembangan sistem informasi dan aplikasi skala kecil hingga besar. Salah satu fakta menarik tentang MySQL adalah bahwa meskipun merupakan perangkat lunak open source, MySQL digunakan oleh banyak perusahaan besar dunia seperti Facebook, Twitter, YouTube, dan Netflix untuk mengelola data dalam skala besar. Keunggulan MySQL terletak pada kemampuannya menangani jutaan transaksi dengan performa tinggi serta kestabilannya yang telah teruji dalam berbagai kondisi.

C. Pengertian DDL

DDL (Data Definition Language) adalah bagian dari bahasa SQL (Structured Query Language) yang digunakan untuk mendefinisikan dan mengelola struktur dari basis data, seperti tabel, indeks, view, dan skema. DDL berfungsi untuk membuat, mengubah, dan menghapus objek-objek dalam database. Perintah-perintah utama dalam DDL meliputi CREATE, ALTER, DROP, dan TRUNCATE. Perintah CREATE digunakan untuk membuat objek baru, seperti membuat tabel dengan struktur kolom dan tipe data tertentu. ALTER digunakan untuk mengubah struktur objek yang sudah ada, misalnya menambahkan kolom baru pada tabel. DROP berfungsi untuk menghapus objek dari database secara permanen, sedangkan TRUNCATE digunakan untuk menghapus seluruh data dalam tabel tanpa menghapus struktur tabelnya. DDL lebih berfokus pada skema atau kerangka kerja dari database. Setiap perintah DDL biasanya dieksekusi secara otomatis (auto-commit), yang berarti bahwa perubahan yang dilakukan akan langsung permanen dan tidak bisa di-rollback. DDL sangat penting dalam tahap perancangan dan pengelolaan basis data, karena menjadi fondasi utama dalam membentuk sistem penyimpanan data yang terstruktur dan efisien (Ery Hartati, 2022).

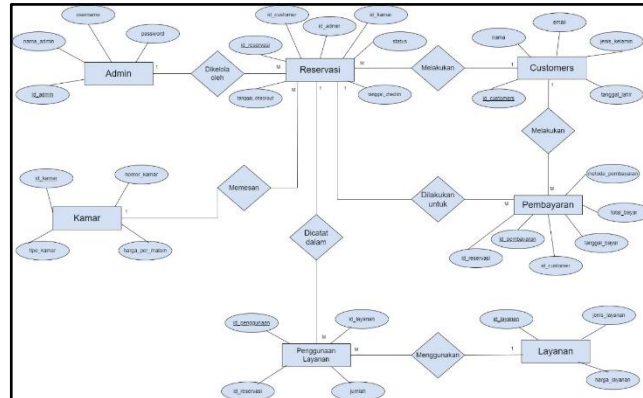
D. Pengertian DCL

DCL (Data Control Language) adalah bagian dari bahasa SQL yang digunakan untuk mengatur hak akses dan izin (privileges) terhadap objek-objek dalam basis data. Tujuan utama dari DCL adalah untuk menjaga keamanan data dan memastikan bahwa hanya pengguna yang memiliki izin tertentu yang dapat melakukan operasi tertentu, seperti membaca, menulis, atau mengubah data. Perintah utama dalam DCL adalah GRANT dan REVOKE. Perintah GRANT digunakan untuk memberikan hak akses kepada pengguna, seperti hak untuk menjalankan perintah SELECT, INSERT, UPDATE, atau DELETE pada tabel tertentu. Sementara itu, REVOKE digunakan untuk mencabut hak akses yang sebelumnya telah diberikan. DCL sangat penting dalam sistem manajemen basis data yang melibatkan banyak pengguna, terutama dalam lingkungan perusahaan atau organisasi besar, karena dapat mengontrol siapa saja yang bisa mengakses data sensitif dan menjaga integritas serta kerahasiaan data.

BAB II

SOAL DAN PEMBAHASAN

1. Buatlah ERD dan database sebuah website manajemen hotel.



Gambar 1.1 ERD

Penjelasan:

ERD (Entity Relationship Diagram) yang ditampilkan menggambarkan sistem reservasi hotel yang terdiri dari beberapa entitas utama, yaitu Customers, Admin, Kamar, Reservasi, Pembayaran, Layanan, dan Penggunaan Layanan. Setiap entitas memiliki atribut yang merepresentasikan data penting dalam sistem. Entitas Customers menyimpan informasi pelanggan seperti nama, email, jenis kelamin, dan tanggal lahir. Seorang pelanggan dapat melakukan banyak reservasi, sehingga relasi antara Customers dan Reservasi adalah satu ke banyak (1:M). Reservasi ini kemudian dikelola oleh Admin, yaitu petugas hotel yang mengelola data reservasi. Karena satu admin dapat menangani banyak reservasi, maka hubungan antara Admin dan Reservasi juga bersifat satu ke banyak (1:M).

Entitas Kamar merepresentasikan data kamar hotel, seperti nomor kamar, tipe kamar, dan harga per malam. Satu kamar bisa dipesan berkali-kali oleh pelanggan yang berbeda di waktu yang berbeda, sehingga hubungan antara Kamar dan Reservasi juga satu ke banyak (1:M). Dalam satu reservasi, pelanggan akan melakukan pembayaran, dan bisa jadi pembayaran dilakukan lebih dari sekali—misalnya membayar DP dan sisanya kemudian—maka hubungan Reservasi dengan Pembayaran juga satu ke banyak (1:M). Begitu juga dengan pelanggan yang bisa saja melakukan beberapa transaksi pembayaran,

menjadikan relasi antara Customers dan Pembayaran adalah satu ke banyak (1:M). Selain kamar, pelanggan juga dapat menggunakan layanan tambahan seperti laundry, room service, dan sebagainya, yang direpresentasikan oleh entitas Layanan. Penggunaan layanan ini dicatat dalam entitas Penggunaan Layanan, yang menjadi jembatan antara Reservasi dan Layanan. Karena satu reservasi bisa mencatat banyak layanan yang digunakan, dan satu layanan bisa digunakan oleh banyak reservasi, maka hubungan Reservasi ke Penggunaan Layanan dan Layanan ke Penggunaan Layanan adalah banyak ke satu (M:1).

```

1 • CREATE DATABASE SI_ManajemenHotel;
2 • USE SI_ManajemenHotel;
3
4 -- 1. Tabel ADMIN
5 • CREATE TABLE admin (
6     id_admin INT AUTO_INCREMENT PRIMARY KEY,
7     nama_admin VARCHAR(100),
8     username VARCHAR(50) UNIQUE,
9     password VARCHAR(100)
10 );
11
12 • INSERT INTO admin (nama_admin, username, password) VALUES
13 ('Dina Purnama', 'dina_admin', 'dina123'),
14 ('Rudi Hartono', 'rudi_admin', 'rudi456'),
15 ('Lina Kusuma', 'lina_admin', 'lina789'),
16 ('Agus Salim', 'agus_admin', 'agus321');

```

Gambar 1.2 database dan tabel admin

Soure code:

CREATE DATABASE SI_ManajemenHotel;

USE SI_ManajemenHotel;

-- 1. Tabel ADMIN

CREATE TABLE admin (

id_admin INT AUTO_INCREMENT PRIMARY KEY,

nama_admin VARCHAR(100),

username VARCHAR(50) UNIQUE,

password VARCHAR(100)

);


```
INSERT INTO admin (nama_admin, username, password) VALUES
```

```
('Dina Purnama', 'dina_admin', 'dina123'),
```

```
('Rudi Hartono', 'rudi_admin', 'rudi456'),
```

```
('Lina Kusuma', 'lina_admin', 'lina789'),
```

```
('Agus Salim', 'agus_admin', 'agus321');
```

Penjelasan code:

Kode di atas adalah serangkaian perintah SQL yang digunakan untuk membuat dan mengelola basis data bernama SI_ManajemenHotel. Perintah pertama, **CREATE DATABASE** SI_ManajemenHotel;, digunakan untuk membuat sebuah database baru dengan nama tersebut. Setelah itu, perintah **USE SI_ManajemenHotel;** dijalankan untuk mulai menggunakan database tersebut sebagai konteks kerja. Selanjutnya, dibuatlah sebuah tabel bernama admin menggunakan perintah **CREATE TABLE admin**, yang memiliki empat kolom yaitu **id_admin**, **nama_admin**, **username**, dan **password**. Kolom **id_admin** bertipe data **INT**, diset sebagai **AUTO_INCREMENT** agar nilainya bertambah otomatis setiap kali data baru ditambahkan, dan dijadikan **PRIMARY KEY** sebagai identitas unik untuk setiap baris. Kolom **nama_admin** menyimpan nama admin dalam format teks hingga 100 karakter, sedangkan **username** dibatasi 50 karakter dan ditetapkan sebagai **UNIQUE** untuk memastikan tidak ada dua admin yang memiliki username sama. Kolom **password** menyimpan kata sandi admin dengan panjang hingga 100 karakter. Setelah struktur tabel dibuat, perintah **INSERT INTO** digunakan untuk menambahkan empat data admin, masing-masing dengan nama, username, dan password yang berbeda.

	id_admin	nama_admin	username	password
▶	1	Dina Purnama	dina_admin	dina123
	2	Rudi Hartono	rudi_admin	rudi456
	3	Lina Kusuma	lina_admin	lina789
	4	Agus Salim	agus_admin	agus321
*	NULL	NULL	NULL	NULL

Gambar 1.3 output tabel admin

```

21 • CREATE TABLE customers (
22     id_customer INT AUTO_INCREMENT PRIMARY KEY,
23     nama VARCHAR(100),
24     email VARCHAR(100),
25     jenis_kelamin ENUM('L', 'P'),
26     tanggal_lahir DATE
27 );

29 • INSERT INTO customers (nama, email, jenis_kelamin, tanggal_lahir) VALUES
30     ('Andi Wijaya', 'andi@gmail.com', 'L', '1990-05-12'),
31     ('Siti Lestari', 'siti@gmail.com', 'P', '1995-08-23'),
32     ('Budi Santoso', 'budi@gmail.com', 'L', '1987-11-04'),
33     ('Dewi Anjani', 'dewi@gmail.com', 'P', '1989-07-15'),
34     ('Rian Saputra', 'rian@gmail.com', 'L', '1993-12-01'),
35     ('Linda Oktaviani', 'linda@gmail.com', 'P', '1996-03-29'),
36     ('Fajar Maulana', 'fajar@gmail.com', 'L', '1985-02-17');

```

Gambar 1.4 sql tabel customers

Source code:

```

CREATE TABLE customers (
    id_customer INT AUTO_INCREMENT PRIMARY KEY,
    nama VARCHAR(100),
    email VARCHAR(100),
    jenis_kelamin ENUM('L', 'P'),
    tanggal_lahir DATE
);

INSERT INTO customers (nama, email, jenis_kelamin, tanggal_lahir) VALUES
('Andi Wijaya', 'andi@gmail.com', 'L', '1990-05-12'),
('Siti Lestari', 'siti@gmail.com', 'P', '1995-08-23'),
('Budi Santoso', 'budi@gmail.com', 'L', '1987-11-04'),
('Dewi Anjani', 'dewi@gmail.com', 'P', '1989-07-15'),
('Rian Saputra', 'rian@gmail.com', 'L', '1993-12-01'),
('Linda Oktaviani', 'linda@gmail.com', 'P', '1996-03-29'),
('Fajar Maulana', 'fajar@gmail.com', 'L', '1985-02-17');

```

Penjelasan:

Kode SQL di atas terdiri dari dua bagian utama: perintah untuk membuat tabel dan perintah untuk memasukkan data ke dalam tabel tersebut. Pertama, perintah **CREATE TABLE** customers digunakan untuk membuat sebuah tabel baru bernama customers dengan lima kolom: **id_customer**, **nama**, **email**, **jenis_kelamin**, dan **tanggal_lahir**. Kolom **id_customer** bertipe INT, diberi atribut

AUTO_INCREMENT agar nilainya bertambah otomatis setiap kali data baru ditambahkan, dan dijadikan sebagai **PRIMARY KEY** untuk memastikan setiap pelanggan memiliki ID unik. Kolom nama dan email bertipe **VARCHAR(100)**, yang berarti dapat menyimpan teks hingga 100 karakter. Kolom jenis_kelamin menggunakan tipe data **ENUM** yang hanya menerima dua nilai, yaitu 'L' (laki-laki) dan 'P' (perempuan). Kolom terakhir, tanggal_lahir, bertipe **DATE** untuk menyimpan tanggal lahir pelanggan. Bagian kedua adalah perintah **INSERT INTO customers**, yang digunakan untuk menambahkan tujuh baris data pelanggan ke dalam tabel. Masing-masing baris berisi informasi nama, email, jenis kelamin, dan tanggal lahir pelanggan. Contohnya, baris pertama memasukkan data pelanggan bernama "Andi Wijaya" dengan email "andi@gmail.com", jenis kelamin "L", dan tanggal lahir "1990-05-12". Data lainnya dimasukkan dengan format serupa.

	id_customer	nama	email	jenis_kelamin	tanggal_lahir
▶	1	Andi Wijaya	andi@gmail.com	L	1990-05-12
	2	Siti Lestari	siti@gmail.com	P	1995-08-23
	3	Budi Santoso	budi@gmail.com	L	1987-11-04
	4	Dewi Anjani	dewi@gmail.com	P	1989-07-15
	5	Rian Saputra	rian@gmail.com	L	1993-12-01
	6	Linda Oktaviani	linda@gmail.com	P	1996-03-29
	7	Fajar Maulana	fajar@gmail.com	L	1985-02-17
*	NULL	NULL	NULL	NULL	NULL

Gambar 1.5 output tabel customers

```

38 CREATE TABLE kamar (
39     id_kamar INT AUTO_INCREMENT PRIMARY KEY,
40     nomor_kamar VARCHAR(10),
41     tipe_kamar ENUM('Single', 'Double', 'Suite'),
42     harga_per_malam DECIMAL(10,2)
43 );
44 • INSERT INTO kamar (nomor_kamar, tipe_kamar, harga_per_malam) VALUES
45 ('101', 'Single', 300000.00),
46 ('102', 'Double', 450000.00),
47 ('201', 'Suite', 750000.00),
48 ('202', 'Single', 320000.00),
49 ('203', 'Double', 470000.00),
50 ('301', 'Suite', 850000.00),
51 ('302', 'Double', 500000.00),
52 ('303', 'Single', 310000.00);

```

Gambar 1.6 sql tabel kamar

Source Code:

CREATE TABLE kamar (

```

id_kamar INT AUTO_INCREMENT PRIMARY KEY,
nomor_kamar VARCHAR(10),
tipe_kamar ENUM('Single', 'Double', 'Suite'),
harga_per_malam DECIMAL(10,2)
);
INSERT INTO kamar (nomor_kamar, tipe_kamar, harga_per_malam) VALUES
('101', 'Single', 300000.00),
('102', 'Double', 450000.00),
('201', 'Suite', 750000.00),
('202', 'Single', 320000.00),
('203', 'Double', 470000.00),
('301', 'Suite', 850000.00),
('302', 'Double', 500000.00),
('303', 'Single', 310000.00);

```

Penjelasan :

Pertama, perintah **CREATE TABLE** kamar digunakan untuk membuat sebuah tabel baru bernama kamar yang berisi informasi mengenai kamar-kamar yang tersedia. Tabel ini memiliki empat kolom utama: **id_kamar**, **nomor_kamar**, **tipe_kamar**, dan **harga_per_malam**. Kolom **id_kamar** bertipe **INT** dan memiliki atribut **AUTO_INCREMENT** serta dijadikan **PRIMARY KEY**, yang artinya nilainya akan bertambah secara otomatis dan digunakan sebagai identifikasi unik untuk setiap kamar. Kolom **nomor_kamar** bertipe **VARCHAR(10)** untuk menyimpan nomor kamar seperti "101" atau "202". Kolom **tipe_kamar** menggunakan tipe **ENUM** yang membatasi nilai yang dapat dimasukkan hanya pada 'Single', 'Double', atau 'Suite'. Kolom **harga_per_malam** bertipe **DECIMAL(10,2)**, digunakan untuk menyimpan harga sewa per malam dengan dua angka desimal.

Bagian kedua dari kode tersebut menggunakan perintah **INSERT INTO** kamar untuk menambahkan delapan baris data kamar ke dalam tabel. Masing-masing baris berisi informasi nomor kamar, tipe kamar, dan harga per malam. Contohnya, kamar dengan nomor "101" bertipe "Single" dan memiliki harga sewa sebesar 300.000,00 per malam.

	id_kamar	nomor_kamar	tipe_kamar	harga_per_malam
▶	1	101	Single	300000.00
	2	102	Double	450000.00
	3	201	Suite	750000.00
	4	202	Single	320000.00
	5	203	Double	470000.00
	6	301	Suite	850000.00
	7	302	Double	500000.00
	8	303	Single	310000.00
✱	NULL	NULL	NULL	NULL

Gambar 1.7 Output tabel kamar

```

55 • CREATE TABLE reservasi (
56     id_reservasi INT AUTO_INCREMENT PRIMARY KEY,
57     id_customer INT,
58     id_kamar INT,
59     id_admin INT,
60     tanggal_checkin DATE,
61     tanggal_checkout DATE,
62     status ENUM('Booked', 'Checked In', 'Cancelled'),
63     FOREIGN KEY (id_customer) REFERENCES customers(id_customer),
64     FOREIGN KEY (id_kamar) REFERENCES kamar(id_kamar),
65     FOREIGN KEY (id_admin) REFERENCES admin(id_admin));
66 • INSERT INTO reservasi (id_customer, id_kamar, id_admin, tanggal_checkin, tanggal_checkout, status) VALUES
67 (1, 1, 1, '2025-05-01', '2025-05-03', 'Checked In'),
68 (2, 2, 2, '2025-05-02', '2025-05-05', 'Booked'),
69 (3, 3, 1, '2025-04-28', '2025-04-30', 'Cancelled'),
70 (4, 4, 3, '2025-05-01', '2025-05-04', 'Checked In'),
71 (5, 5, 4, '2025-05-02', '2025-05-06', 'Booked'),
72 (6, 6, 2, '2025-04-29', '2025-05-01', 'Checked In'),
73 (2, 7, 1, '2025-05-03', '2025-05-05', 'Cancelled'),
74 (1, 8, 4, '2025-05-04', '2025-05-06', 'Booked');

```

Gambar 1.8 sql tabel reservasi

Source code :

```

CREATE TABLE reservasi (
    id_reservasi INT AUTO_INCREMENT PRIMARY KEY,
    id_customer INT,
    id_kamar INT,
    id_admin INT,
    tanggal_checkin DATE,
    tanggal_checkout DATE,
    status ENUM('Booked', 'Checked In', 'Cancelled'),
    FOREIGN KEY (id_customer) REFERENCES customers(id_customer),
    FOREIGN KEY (id_kamar) REFERENCES kamar(id_kamar),

```

```

FOREIGN KEY (id_admin) REFERENCES admin(id_admin)
);
INSERT INTO reservasi (id_customer, id_kamar, id_admin, tanggal_checkin,
tanggal_checkout, status) VALUES
(1, 1, 1, '2025-05-01', '2025-05-03', 'Checked In'),
(2, 2, 2, '2025-05-02', '2025-05-05', 'Booked'),
(3, 3, 1, '2025-04-28', '2025-04-30', 'Cancelled'),
(4, 4, 3, '2025-05-01', '2025-05-04', 'Checked In'),
(5, 5, 4, '2025-05-02', '2025-05-06', 'Booked'),
(6, 6, 2, '2025-04-29', '2025-05-01', 'Checked In'),
(2, 7, 1, '2025-05-03', '2025-05-05', 'Cancelled'),
(1, 8, 4, '2025-05-04', '2025-05-06', 'Booked');

```

Penjelasan :

Bagian pertama adalah perintah **CREATE TABLE** reservasi, yang digunakan untuk membuat tabel baru bernama reservasi. Tabel ini memiliki tujuh kolom: **id_reservasi**, **id_customer**, **id_kamar**, **id_admin**, **tanggal_checkin**, **tanggal_checkout**, dan **status**. Kolom **id_reservasi** bertipe **INT**, diset sebagai **AUTO_INCREMENT** dan **PRIMARY KEY**, yang berarti akan terisi otomatis dan digunakan sebagai identifikasi unik untuk setiap reservasi. Kolom **id_customer**, **id_kamar**, dan **id_admin** masing-masing bertipe **INT** dan berfungsi sebagai foreign key yang merujuk ke tabel customers, kamar, dan admin. Ini menciptakan relasi antar tabel yang memastikan integritas data. Kolom **tanggal_checkin** dan **tanggal_checkout** bertipe **DATE**, menyimpan tanggal pelanggan akan masuk dan keluar. Kolom status menggunakan tipe **ENUM** dengan tiga nilai yang diizinkan: 'Booked', 'Checked In', dan 'Cancelled', untuk menandai status reservasi. Bagian kedua dari kode adalah perintah **INSERT INTO** reservasi, yang berfungsi untuk menambahkan delapan data reservasi ke dalam tabel. Setiap baris menyertakan **ID pelanggan**, **ID kamar**, **ID admin** yang menangani, serta tanggal check-in, check-out, dan status reservasinya. Misalnya, reservasi pertama menunjukkan bahwa pelanggan dengan ID 1 telah memesan kamar dengan ID 1 melalui admin dengan ID 1 untuk tanggal 1 hingga 3 Mei 2025, dengan status "Checked In".

	id_reservasi	id_customer	id_kamar	id_admin	tanggal_checkin	tanggal_checkout	status
▶	1	1	1	1	2025-05-01	2025-05-03	Checked In
	2	2	2	2	2025-05-02	2025-05-05	Booked
	3	3	3	1	2025-04-28	2025-04-30	Cancelled
	4	4	4	3	2025-05-01	2025-05-04	Checked In
	5	5	5	4	2025-05-02	2025-05-06	Booked
	6	6	6	2	2025-04-29	2025-05-01	Checked In
	7	2	7	1	2025-05-03	2025-05-05	Cancelled
	8	1	8	4	2025-05-04	2025-05-06	Booked
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Gambar 1.9 output tabel reservasi

```

80 • CREATE TABLE pembayaran (
81     id_pembayaran INT AUTO_INCREMENT PRIMARY KEY,
82     id_reservasi INT,
83     id_customer INT,
84     metode_pembayaran ENUM('Cash', 'Credit Card', 'Transfer'),
85     total_bayar DECIMAL(12,2),
86     tanggal_bayar DATE,
87     FOREIGN KEY (id_reservasi) REFERENCES reservasi(id_reservasi),
88     FOREIGN KEY (id_customer) REFERENCES customers(id_customer)
89 );
90
91 • INSERT INTO pembayaran (id_reservasi, id_customer, metode_pembayaran, total_bayar, tanggal_bayar) VALUES
92 (1, 1, 'Credit Card', 600000.00, '2025-05-01'),
93 (3, 3, 'Cash', 750000.00, '2025-04-27'),
94 (4, 4, 'Transfer', 960000.00, '2025-05-01'),
95 (5, 5, 'Credit Card', 1880000.00, '2025-05-02'),
96 (6, 6, 'Cash', 1700000.00, '2025-04-29'),
97 (2, 2, 'Cash', 1350000.00, '2025-05-02');

```

Gambar 1.10 sql tabel pembayaran

Source code:

```

CREATE TABLE pembayaran (
    id_pembayaran INT AUTO_INCREMENT PRIMARY KEY,
    id_reservasi INT,
    id_customer INT,
    metode_pembayaran ENUM('Cash', 'Credit Card', 'Transfer'),
    total_bayar DECIMAL(12,2),
    tanggal_bayar DATE,
    FOREIGN KEY (id_reservasi) REFERENCES reservasi(id_reservasi),
    FOREIGN KEY (id_customer) REFERENCES customers(id_customer)
);

INSERT INTO pembayaran (id_reservasi, id_customer, metode_pembayaran,
total_bayar, tanggal_bayar) VALUES

```



```
(1, 1, 'Credit Card', 600000.00, '2025-05-01'),  
(3, 3, 'Cash', 750000.00, '2025-04-27'),  
(4, 4, 'Transfer', 960000.00, '2025-05-01'),  
(5, 5, 'Credit Card', 1880000.00, '2025-05-02'),  
(6, 6, 'Cash', 1700000.00, '2025-04-29'),  
(2, 2, 'Cash', 1350000.00, '2025-05-02');
```

Penjelasan :

Kode SQL di atas digunakan untuk membuat tabel pembayaran dan mengisi data awal ke dalamnya. Bagian pertama adalah perintah **CREATE TABLE pembayaran**, yang bertujuan membentuk tabel bernama pembayaran yang akan menyimpan informasi terkait proses pembayaran reservasi kamar hotel. Tabel ini memiliki enam kolom: **id_pembayaran**, **id_reservasi**, **id_customer**, **metode_pembayaran**, **total_bayar**, dan **tanggal_bayar**. Kolom **id_pembayaran** bertipe **INT**, bersifat **AUTO_INCREMENT**, dan ditetapkan sebagai **PRIMARY KEY** agar setiap pembayaran memiliki ID unik yang dihasilkan secara otomatis. Kolom **id_reservasi** dan **id_customer** merupakan foreign key yang masing-masing merujuk ke tabel reservasi dan customers, sehingga memastikan bahwa setiap pembayaran selalu terkait dengan reservasi dan pelanggan yang valid. Kolom **metode_pembayaran** menggunakan tipe **ENUM** yang hanya menerima tiga pilihan: 'Cash', 'Credit Card', dan 'Transfer', sehingga hanya metode pembayaran yang sah dapat disimpan. Sementara itu, kolom **total_bayar** bertipe **DECIMAL(12,2)** untuk menyimpan jumlah pembayaran secara presisi, dan **tanggal_bayar** bertipe **DATE** untuk mencatat kapan pembayaran dilakukan.

Bagian kedua dari kode merupakan perintah **INSERT INTO pembayaran**, yang berisi enam baris data yang menggambarkan transaksi pembayaran dari beberapa pelanggan. Setiap baris mencatat ID reservasi dan ID pelanggan terkait, metode pembayaran yang digunakan, jumlah total pembayaran, serta tanggal pembayaran dilakukan. Sebagai contoh, data pertama menunjukkan bahwa pelanggan dengan ID 1 membayar reservasi ID 1 menggunakan kartu kredit dengan jumlah sebesar 600.000 rupiah pada tanggal 1 Mei 2025.

	id_pembayaran	id_reservasi	id_customer	metode_pembayaran	total_bayar	tanggal_bayar
▶	1	1	1	Credit Card	600000.00	2025-05-01
	2	3	3	Cash	750000.00	2025-04-27
	3	4	4	Transfer	960000.00	2025-05-01
	4	5	5	Credit Card	1880000.00	2025-05-02
	5	6	6	Cash	1700000.00	2025-04-29
	6	2	2	Cash	1350000.00	2025-05-02
•	NULL	NULL	NULL	NULL	NULL	NULL

Gambar 1.11 Output tabel pembayaran

```

101 • CREATE TABLE layanan (
102     id_layanan INT AUTO_INCREMENT PRIMARY KEY,
103     jenis_layanan VARCHAR(100),
104     harga_layanan DECIMAL(10,2)
105 );
106
107 • INSERT INTO layanan (jenis_layanan, harga_layanan) VALUES
108     ('Laundry', 50000.00),
109     ('Room Service', 75000.00),
110     ('Spa', 120000.00),
111     ('Gym', 30000.00),
112     ('Breakfast Buffet', 100000.00),
113     ('Valet Parking', 40000.00);

```

Gambar 1.12 sql tabel layanan

Source code:

```

CREATE TABLE layanan (
    id_layanan INT AUTO_INCREMENT PRIMARY KEY,
    jenis_layanan VARCHAR(100),
    harga_layanan DECIMAL(10,2)
);

INSERT INTO layanan (jenis_layanan, harga_layanan) VALUES
('Laundry', 50000.00),
('Room Service', 75000.00),
('Spa', 120000.00),
('Gym', 30000.00),
('Breakfast Buffet', 100000.00),
('Valet Parking', 40000.00);

```

Penjelasan :

Pada bagian pertama, perintah **CREATE TABLE layanan** digunakan untuk membuat tabel dengan tiga kolom: **id_layanan**, **jenis_layanan**, dan **harga_layanan**. Kolom **id_layanan** bertipe **INT**, memiliki atribut **AUTO_INCREMENT**, dan dijadikan **PRIMARY KEY**, sehingga setiap layanan akan memiliki ID unik yang bertambah otomatis setiap kali data baru dimasukkan. Kolom **jenis_layanan** bertipe **VARCHAR(100)**, yang memungkinkan penyimpanan nama atau deskripsi layanan hingga 100 karakter. Sedangkan kolom **harga_layanan** bertipe **DECIMAL(10,2)**, yang digunakan untuk mencatat harga layanan dengan dua angka di belakang koma, sehingga cocok untuk keperluan keuangan yang membutuhkan ketelitian nilai. Bagian kedua dari kode berisi perintah **INSERT INTO layanan**, yang digunakan untuk menambahkan enam jenis layanan ke dalam tabel tersebut. Setiap baris memasukkan nama layanan dan harga yang dikenakan. Contohnya, layanan "Laundry" dikenakan biaya sebesar 50.000 rupiah, sementara "Spa" dikenakan biaya 120.000 rupiah. Layanan lainnya mencakup "Room Service", "Gym", "Breakfast Buffet", dan "Valet Parking", masing-masing dengan harga yang berbeda.

	id_layanan	jenis_layanan	harga_layanan
▶	1	Laundry	50000.00
	2	Room Service	75000.00
	3	Spa	120000.00
	4	Gym	30000.00
	5	Breakfast Buffet	100000.00
	6	Valet Parking	40000.00
✱	NULL	NULL	NULL

Gambar 1.13 output tabel layanan

```
116 • CREATE TABLE penggunaan_layanan (  
117     id_penggunaan INT AUTO_INCREMENT PRIMARY KEY,  
118     id_reservasi INT,  
119     id_layanan INT,  
120     jumlah INT,  
121     FOREIGN KEY (id_reservasi) REFERENCES reservasi(id_reservasi),  
122     FOREIGN KEY (id_layanan) REFERENCES layanan(id_layanan)  
123 );
```

```

125 • INSERT INTO penggunaan_layanan (id_reservasi, id_layanan, jumlah) VALUES
126 (1, 1, 2), -- Laundry
127 (1, 2, 1), -- Room Service
128 (3, 3, 1), -- Spa
129 (4, 4, 1), -- Gym
130 (4, 5, 2), -- Breakfast
131 (5, 6, 1), -- Valet
132 (6, 1, 1), -- Laundry
133 (6, 3, 1), -- Spa
134 (2, 2, 2); -- Room Service

```

Gambar 1.14 sql tabel penggunaan layanan

	id_penggunaan	id_reservasi	id_layanan	jumlah
▶	1	1	1	2
	2	1	2	1
	3	3	3	1
	4	4	4	1
	5	4	5	2
	6	5	6	1
	7	6	1	1
	8	6	3	1
	9	2	2	2
✱	NULL	NULL	NULL	NULL

Gambar 1. 15 output tabel penggunaan layanan

```

139 • SELECT r.id_reservasi, c.nama AS nama_customer, k.nomor_kamar, a.nama_admin, r.status
140 FROM reservasi r
141 JOIN customers c ON r.id_customer = c.id_customer
142 JOIN kamar k ON r.id_kamar = k.id_kamar
143 JOIN admin a ON r.id_admin = a.id_admin;

```

Gambar 1. 16 sql Menampilkan daftar reservasi lengkap dengan nama customer, nomor kamar, dan nama admin yang melayani

Source code:

```

SELECT r.id_reservasi, c.nama AS nama_customer, k.nomor_kamar,
a.nama_admin, r.status
FROM reservasi r
JOIN customers c ON r.id_customer = c.id_customer
JOIN kamar k ON r.id_kamar = k.id_kamar
JOIN admin a ON r.id_admin = a.id_admin;

```

Penjelasan:

Kode SQL di atas merupakan sebuah perintah **SELECT** yang digunakan untuk menampilkan data reservasi dengan informasi yang lebih lengkap melalui penggabungan (join) beberapa tabel yang saling terkait. Tabel utama yang digunakan adalah reservasi (diberi alias r), dan digabungkan dengan tabel

lainnya: customers (alias c), kamar (alias k), dan admin (alias a). Penggabungan dilakukan menggunakan JOIN berdasarkan relasi antar-ID, yaitu `r.id_customer = c.id_customer`, `r.id_kamar = k.id_kamar`, dan `r.id_admin = a.id_admin`, untuk memastikan bahwa setiap data reservasi akan disertai informasi dari pelanggan, kamar, dan admin yang menangani.

Kolom-kolom yang diambil dari hasil gabungan ini mencakup `r.id_reservasi` (ID reservasi), `c.nama` yang diubah alias menjadi `nama_customer` (nama pelanggan), `k.nomor_kamar` (nomor kamar yang dipesan), `a.nama_admin` (nama admin yang mengelola reservasi), dan `r.status` (status reservasi seperti "Booked", "Checked In", atau "Cancelled").

Hasil dari query ini adalah sebuah daftar reservasi lengkap yang menyajikan identitas pelanggan, kamar yang digunakan, petugas admin yang bertugas, serta status dari reservasi tersebut.

	id_reservasi	nama_customer	nomor_kamar	nama_admin	status
▶	1	Andi Wijaya	101	Dina Purnama	Checked In
	3	Budi Santoso	201	Dina Purnama	Cancelled
	7	Siti Lestari	302	Dina Purnama	Cancelled
	2	Siti Lestari	102	Rudi Hartono	Booked
	6	Linda Oktaviani	301	Rudi Hartono	Checked In
	4	Dewi Anjani	202	Lina Kusuma	Checked In
	5	Rian Saputra	203	Agus Salim	Booked
	8	Andi Wijaya	303	Agus Salim	Booked

Gambar 1. 17 output yang dihasilkan

```

147 • SELECT
148     c.nama AS customer_name,
149     k.nomor_kamar AS room_number,
150     r.tanggal_checkin AS checkin_date,
151     r.tanggal_checkout AS checkout_date,
152     r.status AS reservation_status
153 FROM reservasi r
154 JOIN customers c ON r.id_customer = c.id_customer
155 JOIN kamar k ON r.id_kamar = k.id_kamar
156 WHERE r.status = 'Booked' AND k.tipe_kamar = 'Double';

```

Gambar 1. 18 Menampilkan semua pelanggan yang memesan kamar tipe "Double" dan yang status reservasinya "Booked"

Source code:

```

SELECT
    c.nama AS customer_name,
    k.nomor_kamar AS room_number,

```

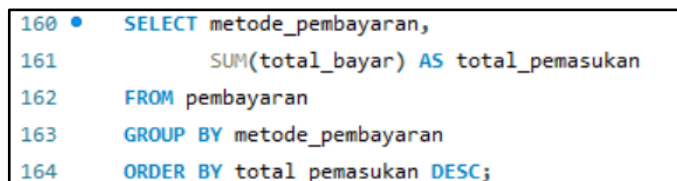
```

r.tanggal_checkin AS checkin_date,
r.tanggal_checkout AS checkout_date,
r.status AS reservation_status
FROM reservasi r
JOIN customers c ON r.id_customer = c.id_customer
JOIN kamar k ON r.id_kamar = k.id_kamar
WHERE r.status = 'Booked' AND k.tipe_kamar = 'Double';

```

Penjelasan :

Kode SQL di atas merupakan perintah **SELECT** yang digunakan untuk menampilkan daftar reservasi kamar hotel dengan kondisi tertentu, yaitu reservasi yang berstatus "Booked" dan menggunakan kamar bertipe "Double". Query ini mengambil data dari tabel reservasi (diberi alias r), lalu melakukan penggabungan dengan tabel customers (c) melalui relasi **r.id_customer = c.id_customer** untuk mendapatkan informasi pelanggan, serta dengan tabel kamar (k) melalui **r.id_kamar = k.id_kamar** untuk memperoleh detail kamar yang dipesan. Kolom-kolom yang ditampilkan dalam hasil query ini meliputi c.nama yang diberi alias customer_name (nama pelanggan), k.nomor_kamar sebagai room_number (nomor kamar), r.tanggal_checkin sebagai checkin_date, r.tanggal_checkout sebagai checkout_date, dan r.status sebagai reservation_status. Kondisi pada klausa **WHERE** memastikan bahwa hanya data reservasi yang sedang dalam status Booked dan berkaitan dengan kamar tipe Double saja yang diambil.



```

160 • SELECT metode_pembayaran,
161       SUM(total_bayar) AS total_pemasukan
162 FROM pembayaran
163 GROUP BY metode_pembayaran
164 ORDER BY total_pemasukan DESC;

```

Gambar 1.19 sql metode pembayaran

Source code:

```

SELECT metode_pembayaran,
       SUM(total_bayar) AS total_pemasukan
FROM pembayaran
GROUP BY metode_pembayaran
ORDER BY total_pemasukan DESC;

```

Penjelasan :

Kode SQL di atas digunakan untuk melakukan rekapitulasi pemasukan berdasarkan metode pembayaran yang digunakan oleh pelanggan hotel. Dalam query ini, tabel pembayaran dijadikan sumber data, dan kolom metode_pembayaran dipilih untuk dikelompokkan menggunakan klausa **GROUP BY**. Fungsi agregat **SUM(total_bayar)** digunakan untuk menghitung total pemasukan dari setiap metode pembayaran, seperti Cash, Credit Card, dan Transfer. Hasil dari perhitungan tersebut diberi alias total_pemasukan. Selanjutnya, klausa **ORDER BY total_pemasukan DESC** digunakan untuk mengurutkan hasil dari yang paling besar ke yang paling kecil, sehingga metode pembayaran dengan kontribusi pemasukan tertinggi akan muncul di urutan pertama.

	metode_pembayaran	total_pemasukan
▶	Cash	3800000.00
	Credit Card	2480000.00
	Transfer	960000.00

Gambar 1.20 output yang dihasilkan

```
168 • CREATE VIEW view_reservasi_aktif AS
169 SELECT r.id_reservasi, c.nama, k.nomor_kamar, r.tanggal_checkin, r.tanggal_checkout
170 FROM reservasi r
171 JOIN customers c ON r.id_customer = c.id_customer
172 JOIN kamar k ON r.id_kamar = k.id_kamar
173 WHERE r.status = 'Checked In';
```

Gambar 1.21 sql view reservasi aktif

Source code:

```
CREATE VIEW view_reservasi_aktif AS
SELECT r.id_reservasi, c.nama, k.nomor_kamar, r.tanggal_checkin,
r.tanggal_checkout
FROM reservasi r
JOIN customers c ON r.id_customer = c.id_customer
JOIN kamar k ON r.id_kamar = k.id_kamar
WHERE r.status = 'Checked In';
```

Penjelasan :

Kode SQL di atas digunakan untuk membuat sebuah view atau tampilan virtual bernama **view_reservasi_aktif**, yang menyajikan data reservasi yang saat ini sedang aktif, yaitu yang berstatus 'Checked In'. View ini dibentuk dari hasil query **SELECT** yang mengambil data dari tabel reservasi (diberi alias **r**), lalu menggabungkannya dengan tabel customers (**c**) melalui kolom **id_customer**, serta dengan tabel kamar (**k**) melalui kolom **id_kamar**. Data yang ditampilkan dalam view ini mencakup ID reservasi (**r.id_reservasi**), nama pelanggan (**c.nama**), nomor kamar yang digunakan (**k.nomor_kamar**), serta tanggal check-in dan check-out (**r.tanggal_checkin** dan **r.tanggal_checkout**). Filter **WHERE r.status = 'Checked In'** memastikan hanya reservasi yang sudah aktif (tamu sudah check-in) yang dimasukkan ke dalam view ini.

	id_reservasi	nama	nomor_kamar	tanggal_checkin	tanggal_checkout
▶	1	Andi Wijaya	101	2025-05-01	2025-05-03
	4	Dewi Anjani	202	2025-05-01	2025-05-04
	6	Linda Oktaviani	301	2025-04-29	2025-05-01

Gambar 1.22 output yang dihasilkan

```
178 DELIMITER //
179
180 CREATE PROCEDURE tambah_pembayaran (
181     IN p_id_reservasi INT,
182     IN p_id_customer INT,
183     IN p_metode ENUM('Cash', 'Credit Card', 'Transfer'),
184     IN p_total DECIMAL(12,2),
185     IN p_tanggal DATE
186 )
187 BEGIN
188     INSERT INTO pembayaran (id_reservasi, id_customer, metode_pembayaran, total_bayar, tanggal_bayar)
189     VALUES (p_id_reservasi, p_id_customer, p_metode, p_total, p_tanggal);
190 END //
191
192 DELIMITER ;
```

Gambar 1.23 sql store procude

Source code:

DELIMITER //

CREATE PROCEDURE tambah_pembayaran (
 IN p_id_reservasi INT,
 IN p_id_customer INT,

```

    IN p_metode ENUM('Cash','Credit Card','Transfer'),
    IN p_total DECIMAL(12,2),
    IN p_tanggal DATE
)
BEGIN
    INSERT INTO pembayaran (id_reservasi, id_customer,
metode_pembayaran, total_bayar, tanggal_bayar)
    VALUES (p_id_reservasi, p_id_customer, p_metode, p_total, p_tanggal);
END //

DELIMITER ;

```

Penjelasan :

Prosedur `tambah_pembayaran` menerima lima parameter input (ditandai dengan IN): `p_id_reservasi` (ID reservasi), `p_id_customer` (ID pelanggan), `p_metode` (metode pembayaran yang hanya boleh bernilai 'Cash', 'Credit Card', atau 'Transfer'), `p_total` (jumlah pembayaran), dan `p_tanggal` (tanggal pembayaran). Di dalam blok `BEGIN ... END`, prosedur menjalankan perintah `INSERT INTO` `pembayaran`, yang akan menyimpan data ke dalam tabel `pembayaran` menggunakan nilai dari kelima parameter tersebut. Stored procedure ini sangat berguna untuk menyederhanakan proses penambahan data pembayaran, terutama jika prosedur ini dipanggil secara rutin dari aplikasi atau sistem backend hotel.

```

197     SELECT nama,
198           (SELECT SUM(total_bayar)
199            FROM pembayaran
200            WHERE id_customer = c.id_customer) AS total_pembayaran
201     FROM customers c
202     WHERE (SELECT SUM(total_bayar)
203            FROM pembayaran
204            WHERE id_customer = c.id_customer) > (
205            SELECT AVG(total_bayar) FROM pembayaran
206     );

```

Gambar 1.24 sql nested query

Source code:

```
SELECT nama,  
       (SELECT SUM(total_bayar)  
        FROM pembayaran  
        WHERE id_customer = c.id_customer) AS total_pembayaran  
FROM customers c  
WHERE (SELECT SUM(total_bayar)  
       FROM pembayaran  
       WHERE id_customer = c.id_customer) > (  
       SELECT AVG(total_bayar) FROM pembayaran  
       );
```

Penjelasan :

Kode SQL di atas digunakan untuk menampilkan daftar pelanggan (customers) beserta total pembayaran mereka, hanya jika total pembayaran tersebut lebih besar dari rata-rata pembayaran semua pelanggan. Query ini menggunakan subquery bersarang (nested subquery) baik di bagian kolom yang ditampilkan maupun pada klausa **WHERE**.

Pada bagian **SELECT**, kolom nama diambil dari tabel customers (dengan alias c), dan diikuti oleh subquery (**SELECT SUM(total_bayar) FROM pembayaran WHERE id_customer = c.id_customer**) yang menghitung total pembayaran dari masing-masing pelanggan berdasarkan ID mereka. Hasil dari subquery tersebut diberi alias total_pembayaran. Kemudian, klausa **WHERE** digunakan untuk memfilter hanya pelanggan yang total pembayarannya lebih besar dari nilai rata-rata pembayaran secara keseluruhan. Perbandingan ini dilakukan dengan subquery tambahan (**SELECT AVG(total_bayar) FROM pembayaran**), yang menghitung nilai rata-rata dari seluruh kolom total_bayar dalam tabel pembayaran.

	nama	total_pembayaran
▶	Siti Lestari	1350000.00
	Rian Saputra	1880000.00
	Linda Oktaviani	1700000.00

Gambar 1.25 sql output yang dihasilkan

```

210 • CREATE TABLE log_layanan (
211     id_log INT AUTO_INCREMENT PRIMARY KEY,
212     id_penggunaan INT,
213     waktu_log TIMESTAMP DEFAULT CURRENT_TIMESTAMP
214 );

```

Gambar 1.26 sql tabel log layanan

```

216 DELIMITER //
217
218 • CREATE TRIGGER after_penggunaan_layanan
219 AFTER INSERT ON penggunaan_layanan
220 FOR EACH ROW
221 BEGIN
222     INSERT INTO log_layanan (id_penggunaan)
223     VALUES (NEW.id_penggunaan);
224 END //
225
226 DELIMITER ;

```

Gambar 1.27 sql trigger

Source code:

```

DELIMITER //

CREATE TRIGGER after_penggunaan_layanan
AFTER INSERT ON penggunaan_layanan
FOR EACH ROW
BEGIN
    INSERT INTO log_layanan (id_penggunaan)
    VALUES (NEW.id_penggunaan);
END //

DELIMITER ;

```

Penjelasan :

Kode SQL di atas merupakan perintah untuk membuat sebuah trigger bernama `after_penggunaan_layanan` di MySQL. Trigger ini akan secara otomatis dijalankan setelah (AFTER) terjadi penambahan data (INSERT) ke dalam tabel `penggunaan_layanan`. Blok kode dibuka dan ditutup dengan

pengubah delimiter (**DELIMITER //**) agar SQL dapat memproses perintah kompleks yang mengandung titik koma (;) di dalamnya tanpa konflik.

Trigger ini dirancang untuk berjalan per baris (**FOR EACH ROW**), artinya setiap kali satu baris data baru dimasukkan ke dalam **penggunaan_layanan**, trigger ini akan dieksekusi sekali. Isi dari trigger tersebut adalah sebuah perintah **INSERT** yang akan mencatat ID dari data penggunaan layanan yang baru saja dimasukkan (**NEW.id_penggunaan**) ke dalam tabel **log_layanan**. Dengan kata lain, setiap kali ada layanan yang digunakan oleh pelanggan dan dicatat dalam tabel **penggunaan_layanan**, trigger ini otomatis mencatat referensinya di tabel **log_layanan**.

```
230 CREATE USER 'resepsionis'@'localhost' IDENTIFIED BY 'pass123';  
231
```

Gambar 1.28 DDL tambah user

```
233 • GRANT CREATE ON SI_ManajemenHotel.* TO 'resepsionis'@'localhost';  
234
```

Gambar 1.29 DDL Memberi akses create

```
236 • GRANT SELECT, INSERT ON SI_ManajemenHotel.reservasi TO 'resepsionis'@'localhost';  
237
```

Gambar 1.30 DDL Memberi akses data ke tabel 'reservasi'

Source code:

```
CREATE USER 'resepsionis'@'localhost' IDENTIFIED BY 'pass123';  
GRANT CREATE ON SI_ManajemenHotel.* TO 'resepsionis'@'localhost';  
GRANT SELECT, INSERT ON SI_ManajemenHotel.reservasi TO  
'resepsionis'@'localhost';
```

Penjelasan :

Pertama, perintah **CREATE USER 'resepsionis'@'localhost' IDENTIFIED BY 'pass123';** digunakan untuk membuat akun pengguna baru bernama **resepsionis** yang hanya bisa mengakses database dari komputer lokal (**localhost**), dengan kata sandi **pass123**. Selanjutnya, perintah **GRANT CREATE ON SI_ManajemenHotel.* TO 'resepsionis'@'localhost';** memberikan hak akses **CREATE** kepada user tersebut. Kemudian, perintah terakhir **GRANT SELECT, INSERT ON SI_ManajemenHotel.reservasi TO 'resepsionis'@'localhost';** memberikan hak akses membaca data (**SELECT**) dan menambahkan data (**INSERT**) secara spesifik hanya pada tabel **reservasi** di dalam database tersebut.

BAB III

KESIMPULAN DAN SARAN

A. Kesimpulan

Basis data merupakan elemen krusial dalam sistem informasi modern karena memungkinkan penyimpanan, pengelolaan, dan pengambilan data secara terstruktur, aman, dan efisien. Dalam industri perhotelan, penggunaan basis data sangat penting untuk menunjang berbagai proses operasional seperti pemesanan kamar, pencatatan pembayaran, pengelolaan layanan, dan pelaporan administrasi. Sistem database *SI_ManajemenHotel* yang dirancang menggunakan SQL telah mencerminkan penerapan konsep basis data secara menyeluruh dan relasional. Setiap tabel saling terhubung melalui relasi yang jelas dan konsisten menggunakan *foreign key*, sehingga integritas data terjaga.

Selain struktur data yang lengkap, sistem ini juga dilengkapi dengan fitur-fitur lanjutan seperti *JOIN* untuk penggabungan informasi, *view* untuk menampilkan data dinamis, *stored procedure* untuk otomatisasi proses, *trigger* untuk pencatatan log otomatis, serta *nested query* untuk analisis data yang lebih dalam. Sistem juga telah menerapkan kontrol akses melalui DCL, menandakan bahwa aspek keamanan dan hak pengguna turut diperhatikan. Dengan desain yang menyeluruh dan fungsional, sistem ini memberikan fondasi yang kuat untuk pengelolaan hotel secara digital dan dapat diandalkan dalam skala operasional nyata. Database tidak hanya berfungsi sebagai tempat penyimpanan data, tetapi juga sebagai fondasi utama dalam pengambilan keputusan yang cepat dan tepat dalam suatu sistem informasi.

B. Saran

Untuk pengembangan lebih lanjut, disarankan agar sistem ini dilengkapi dengan antarmuka pengguna berbasis aplikasi atau web guna memudahkan operasional harian oleh staf non-teknis. Selain itu, implementasi validasi input pada sisi aplikasi perlu dilakukan untuk mencegah kesalahan data. Menambahkan fitur pencatatan riwayat perubahan data (audit trail) dan mekanisme backup otomatis juga penting sebagai langkah pengamanan data. Dengan pengembangan berkelanjutan, sistem ini dapat menjadi fondasi yang kuat bagi digitalisasi layanan perhotelan secara menyeluruh.

DAFTAR PUSTAKA

- Hastanti, R. P., & Purnama, B. E. (2015). Sistem penjualan berbasis web (e-commerce) pada tata distro kabupaten pacitan. *Bianglala Informatika*, 3(2).
- Kristanto, H. 2004. Konsep dan Perancangan Database. Yogyakarta: Andi Sidik, B. 2005. MySQL Untuk Pengguna, Administrator, dan Pengembangan Aplikasi Web. Bandung: Informatika
- Peranginangin, K. (2006). Aplikasi WEB dengan PHP dan MySQL." Yogyakarta. *Andi Offset*.
- Swara, G. Y., Kom, M., & Pebriadi, Y. (2016). Rekayasa perangkat lunak pemesanan tiket bioskop berbasis web. *Jurnal Teknoif Teknik Informatika Institut Teknologi Padang*, 4(2), 27-39.
- Ery Hartati (2022) 'Sistem Informasi Transaksi Gudang Berbasis Website Pada Cv. Asyura', *Klik - Jurnal Ilmu Komputer*, 3(1), pp. 12–18. Available at: <https://doi.org/10.56869/klik.v3i1.323>.

LAMPIRAN

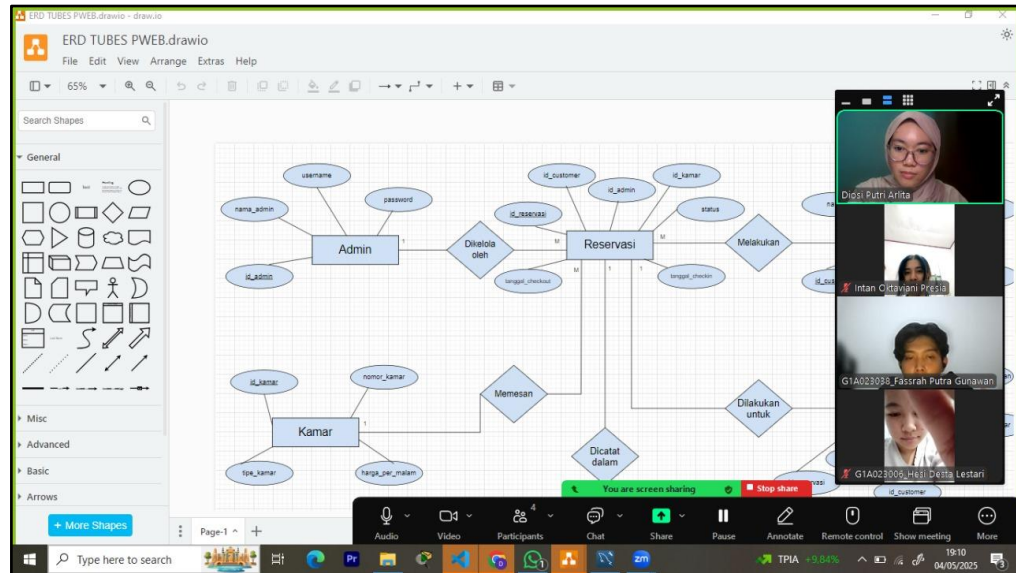
A. Jobdesk

ERD: Diosi Putri Arlita

Database: Fassrah Putra Gunawan

Laporan & PPT: Hesi Desta Lestari dan Intan Oktaviani Presia

B. Dokumentasi





**KEMENTERIAN PENDIDIKAN, KEBUDAYAAN
RISET DAN TEKNOLOGI
UNIVERSITAS BENGKULU
FAKULTAS TEKNIK
PROGRAM STUDI INFORMATIKA**

Jl. Wr. Supratman Kandang Limun, Bengkulu Bengkulu
38371 A Telp: (0736) 344087, 22105 - 227

LEMBAR ASISTENSI TUGAS BESAR

PROYEK BASIS DATA

Nama Mahasiswa : 1. Hesi Desta Lestari (G1A023006)
2. Diosi Putri Arlita (G1A023012)
3. Intan Oktaviani Presia (G1A023014)
4. Fassrah Putra Gunawan (G1A023038)

Dosen : 1. Dr., Endina Putri Puwandari, S.T., M.Kom
2. Tiara Eka Putri, S.T., M.Kom.

Asisten Dosen : 1. Merly Yuni Purnama (G1A022006)
2. Reksi Hendra Pratama (G1A022032)
3. Sinta Ezra Wati Gulo (G1A022040)
4. Fadlan Dwi Febrio (G1A022051)
5. Torang Four Yones Manullang (G1A022052)
6. Wahyu Ozorah Manurung (G1A022060)
7. Shalaudin Muhammad Sah (G1A022070)
8. Dian Ardiyanti Saputri (G1A022084)

Laporan Praktikum

Catatan dan Tanda Tangan

Tugas Besar