

**LAPORAN TUGAS BESAR
PROYEK BASIS DATA LANJUT**



Disusun Oleh :

- | | |
|-------------------------------|-------------|
| 1. Rayhan Muhammad Adha | (G1A023051) |
| 2. Ayu Dewanti Suci | (G1A023057) |
| 3. Mohamad Irvan Ramadhansyah | (G1A023089) |
| 4. Hendro Paulus Limbong | (G1A023091) |

Nama Asisten Dosen :

- | | |
|--------------------------------|-------------|
| 1. Reksi Hendra Pratama | (G1A022032) |
| 2. Merly Yuni Purnama | (G1A022006) |
| 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 Purwandari, S.T., M.Kom.
2. Ir. Tiara Eka Putri, S.T., M.Kom.

**PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS BENGKULU**

2025

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas segala rahmat, hidayah, dan karunia-Nya yang senantiasa melimpahkan keberkahan dalam setiap langkah perjalanan hidup kami. Atas izin-Nya pula, kami dapat menyelesaikan penyusunan laporan tugas besar ini dengan lancar. Penyusunan laporan ini merupakan bagian dari proses akademik yang kami jalani, sekaligus sebagai bentuk tanggung jawab dalam mengerjakan tugas akhir praktikum yang telah diberikan.

Laporan ini merupakan hasil dari proses pembelajaran yang intensif dan penuh dedikasi selama mengikuti praktikum, yang tidak hanya memberikan pengetahuan teoritis, tetapi juga pengalaman praktis yang berharga. Dengan penuh kebanggaan dan antusiasme, kami dengan rendah hati mempersembahkan laporan tugas besar ini. Laporan ini berisi proyek pembuatan sistem manajemen simpan pinjam koperasi. Laporan ini disusun untuk memenuhi tugas mata kuliah Proyek Basis Data Lanjut. Kami mengucapkan terima kasih kepada Abang Wahyu dan Abang Torang selaku asisten dosen yang telah memberikan bimbingan, baik secara moral maupun materi, sehingga kami dapat menyelesaikan laporan ini dengan baik. Ucapan terima kasih juga kami sampaikan kepada teman-teman yang turut membantu dalam proses penyusunan laporan ini.

Akhir kata, kami menyadari bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang membangun dari para pembaca demi perbaikan dan pengembangan di masa mendatang. Semoga laporan ini bermanfaat bagi semua pihak yang terlibat dan dapat menjadi inspirasi bagi karya-karya yang lebih baik ke depannya.

Bengkulu, Mei 2025

Kelompok 4

DAFTAR ISI	
KATA PENGANTAR.....	i
DAFTAR ISI.....	ii
DAFTAR GAMBAR.....	iii
DAFTAR TABEL	iv
BAB I.....	1
PENDAHULUAN.....	1
BAB II	3
LANDASAN TEORI.....	3
BAB III.....	6
HASIL DAN PEMBAHASAN	6
BAB IV	54
PENUTUP.....	54
DAFTAR PUSTAKA	55
Dokumentasi Pengerjaan	56

DAFTAR GAMBAR

Gambar 1. ERD	6
Gambar 2. Source Code Database dan Tabel	8
Gambar 3. Source Code Tabel (1).....	7
Gambar 4. Input Data	14
Gambar 22. Join	19
Gambar 24. Alias dan Operator	23
Gambar 26. Function, Grouping, Sorting.....	27
Gambar 32. View dan Stored Proecedure	28
Gambar 50. Nested Query dan Trigger	45
Gambar 54. DCL	48
Gambar 55. Admin dan User	50
Gambar 56. User	51
Gambar 57. Halaman User	51
Gambar 58. User Mencoba Menghapus Database	51
Gambar 59. Admin.....	52
Gambar 60. Halaman Admin (1).....	52
Gambar 61. Admin Berhasil Menghapus Database	52

DAFTAR TABEL

Gambar 5. Tabel Anggota.....	17
Gambar 6. Tabel Angsuran	17
Gambar 7. Tabel Jenis Simpanan.....	17
Gambar 8. Tabel Jenis Transaksi	18
Gambar 9. Tabel Login	18
Gambar 10. Tabel Notif	18
Gambar 11. Tabel Petugas	18
Gambar 12. Tabel Pinjaman.....	18
Gambar 13. Tabel Simpanan.....	19
Gambar 14. Tabel Transaksi	19
Gambar 22. Tabel Left Join	25
Gambar 25. Tabel Right Join	27
Gambar 26. Tabel Output Inner Join	27
Gambar 27. Tabel Output Full Join	27
Gambar 29. Tabel Output Function	28
Gambar 30. Tabel Output Grouping	28
Gambar 31. Tabel Output Sorting.....	29
Gambar 32. Tabel Output View	30
Gambar 39. Tabel Output Stored Procedure	31
Gambar 50. Tabel Output Nested Query.....	45
Gambar 51. Tabel Output Trigger.....	45
Gambar 53. Output Insert Trigger.....	48

BAB I

PENDAHULUAN

1.1 Latar Belakang

Koperasi merupakan lembaga ekonomi yang berperan penting dalam meningkatkan kesejahteraan anggotanya melalui usaha bersama, salah satunya adalah kegiatan simpan pinjam. Sistem simpan pinjam koperasi memungkinkan anggota untuk menyimpan dana dan memperoleh pinjaman dengan bunga yang relatif rendah dibandingkan lembaga keuangan konvensional. Namun, dalam praktiknya, banyak koperasi masih menggunakan metode manual atau sistem yang kurang terintegrasi dalam mengelola simpan pinjam. Hal ini menyebabkan berbagai kendala seperti pencatatan data yang tidak akurat, proses pelayanan yang lambat, serta kesulitan dalam pelaporan dan pengawasan keuangan.

Kegiatan manajemen simpan pinjam yang melibatkan banyak transaksi, perhitungan bunga, jatuh tempo, hingga laporan keuangan memerlukan sistem yang terstruktur dan efisien. Oleh karena itu, dibutuhkan sebuah sistem informasi yang dapat mengelola seluruh proses tersebut secara otomatis, akurat, dan mudah digunakan. Dengan menerapkan sistem manajemen simpan pinjam koperasi yang terkomputerisasi, koperasi dapat meningkatkan efisiensi operasional, akurasi data, dan transparansi dalam pengelolaan keuangan. Sistem ini juga memungkinkan pengelolaan data anggota, transaksi simpanan, dan pinjaman secara *real-time* serta menyediakan laporan yang mudah diakses bagi pengurus dan anggota.

Oleh karena itu, pengembangan sistem manajemen simpan pinjam koperasi merupakan langkah strategis untuk mendukung keberlangsungan dan perkembangan koperasi dalam memberikan pelayanan terbaik bagi anggotanya. Sistem ini diharapkan dapat membantu koperasi dalam mengoptimalkan pengelolaan keuangan serta meningkatkan partisipasi anggota. Dengan penerapan sistem ini, koperasi dapat meningkatkan profesionalisme layanan, mengurangi beban kerja administratif, serta meningkatkan kepercayaan anggota terhadap pengelolaan dana. Selain itu, sistem ini juga mendukung digitalisasi koperasi dalam menghadapi perkembangan teknologi informasi dan kebutuhan akan layanan yang cepat dan transparan.

1.2 Rumusan Masalah

1. Bagaimana merancang sistem informasi yang dapat mengelola kegiatan simpan pinjam koperasi secara efisien dan terstruktur?
2. Bagaimana sistem dapat memberikan kemudahan bagi pengurus dan anggota dalam mengakses data simpanan dan pinjaman?
3. Bagaimana sistem dapat meningkatkan akurasi dan kecepatan dalam pengolahan data sehingga meminimalisir kesalahan yang terjadi pada sistem manual?

1.3 Tujuan Penulisan

1. Untuk merancang dan membangun sistem informasi yang mendukung proses simpan pinjam koperasi secara digital dan terintegrasi.
2. Menyediakan sistem yang mampu menghasilkan laporan keuangan yang akurat dan dapat diakses dengan mudah oleh semua pihak terkait.
3. Mengembangkan sistem manajemen simpan pinjam koperasi yang terintegrasi secara digital untuk memudahkan proses pencatatan, pengelolaan, dan pelaporan transaksi simpanan dan pinjaman.
4. Meningkatkan transparansi dan akuntabilitas pengelolaan simpan pinjam koperasi guna meningkatkan kepercayaan anggota terhadap koperasi.
5. Menyediakan fitur pelaporan keuangan dan transaksi yang dapat diakses secara *real-time* untuk mendukung pengambilan keputusan yang lebih baik.

1.4 Manfaat Pengembangan Proyek

1. Bagi Koperasi: Mempercepat proses administrasi, mengurangi kesalahan pencatatan, dan mempermudah pengelolaan data keuangan sehingga operasional koperasi menjadi lebih efisien dan terorganisir.
2. Bagi anggota: Mendapatkan pelayanan simpan pinjam yang lebih cepat, transparan, dan terpercaya dengan akses informasi yang jelas mengenai status simpanan dan pinjaman mereka.
3. Bagi Pengurus: Memudahkan monitoring dan evaluasi kinerja koperasi melalui laporan yang akurat dan *real-time*, sehingga pengambilan keputusan menjadi lebih tepat dan strategis.
4. Bagi Pengembangan Koperasi: Mendorong penggunaan teknologi informasi dalam pengelolaan koperasi sebagai langkah modernisasi yang dapat meningkatkan daya saing dan keberlanjutan koperasi di masa depan.

BAB II

LANDASAN TEORI

2.1 Koperasi

Koperasi merupakan bagian dari perekonomian nasional, baik sebagai organisasi ekonomi maupun sebagai gerakan ekonomi. Dalam perkembangannya, koperasi berperan sebagai pendukung perekonomian nasional, dengan jaringan usaha yang kuat dan berdaya saing untuk mengantisipasi berbagai peluang dan tantangan ke depan. Koperasi sebagai bentuk organisasi penting untuk mendorong pertumbuhan ekonomi. Perhimpunan simpan pinjam dan serikat simpan pinjam merupakan sarana alternatif bagi anggota untuk mengumpulkan dana guna meningkatkan taraf hidup, memenuhi kebutuhan sehari-hari, dan mengembangkan usahanya (Hasan et al., 2023).

Koperasi Simpan Pinjam yang selanjutnya disingkat KSP adalah Koperasi yang kegiatannya hanya usaha simpan pinjam. Unit Simpan Pinjam pada Koperasi yang selanjutnya disingkat USP adalah unit koperasi yang bergerak dibidang usaha simpan pinjam, sebagai bagian dari kegiatan usaha koperasi yang bersangkutan. Perkembangan usaha KSP sangat penting dalam mengantisipasi kebutuhan kredit para pelaku usaha kecil dan menengah yang berada di sektor informal. Sehingga penting diidentifikasi keadaan perkembangan usaha KSP tersebut agar dapat diambil kebijakan yang tepat untuk meningkatkan peran KSP (Juswadi & Sumarna, 2023).

2.2 Basis Data

Basis data adalah tempat berkumpulnya data yang saling berhubungan dalam suatu wadah (perusahaan/organisasi) bertujuan agar dapat mempermudah dan mempercepat untuk pemanggilan atau pemanfaatan kembali data tersebut. Basis data (*database*) adalah kumpulan data yang terstruktur secara sistematis dan tersimpan di suatu tempat agar dapat diakses, dimanipulasi, dan dikelola dengan mudah. Basis data digunakan untuk menyimpan informasi yang terkait satu sama lain dan dapat diakses oleh sistem atau aplikasi perangkat lunak. Basis data mengonsolidasi banyak catatan yang sebelumnya disimpan dalam file terpisah. (Wijaya et al., 2021).

2.3 Data Manipulation Language (DML)

Data Manipulation Language (DML) merupakan bahasa basis data yang dipergunakan untuk melakukan modifikasi dan *retrieve* (pengambilan) data pada suatu basis data. *Data Manipulation Language* (DML) adalah set (kumpulan) statemen yang digunakan sebagai perintah untuk mengelola data dalam sebuah tabel. DML yang sering digunakan dan terkenal adalah *Structured Query Language* (SQL) yang digunakan untuk mengambil dan memanipulasi data dalam *database* relasional. SQL terdiri dari sintaks sederhana dalam bentuk instruksi-instruksi dalam melakukan manipulasi data, instruksi tersebut sering disebut dengan query. DML memiliki beberapa perintah utama, seperti *select*, *insert*, *update* dan *delete*. (Setiyadi & Nidaul Khasanah, 2019).

2.4 Data Definition Language (DDL)

Data Definition Language (DDL) adalah kumpulan perintah SQL yang digunakan untuk membangun *database*, bisa juga dikatakan DDL adalah perintah untuk membuat struktur dasar data *database* dan tabel. Perintah-perintah yang termasuk DDL adalah *statement connect*, *create*, *show*, *describe*, *drop*, *rename* dan *alter*. DDL adalah bagian dari SQL yang fokus pada struktur objek dalam basis data. Ini memungkinkan pengguna untuk membuat, mengubah, dan menghapus struktur basis data seperti tabel, indeks, *view*, dan lainnya. DDL juga mengatur aturan integritas, batasan, dan hak akses terhadap objek-objek ini (Sinata, 2023).

2.5 Data Control Language (DCL)

Data Control Language (DCL) adalah bagian dari SQL (*Structured Query Language*) yang digunakan untuk mengatur hak akses (*privilege*) dan kontrol keamanan terhadap data dalam sistem basis data. DCL berfungsi untuk menentukan siapa saja yang diizinkan untuk melakukan tindakan tertentu terhadap objek-objek *database* seperti tabel, *view*, atau prosedur. DCL adalah perintah yang digunakan untuk keperluan keamanan *database* dengan membuat hak akses tertentu bagi setiap *user*. DCL sangat penting dalam sistem yang melibatkan banyak pengguna (*multi-user*), karena membantu menjaga integritas dan kerahasiaan data. *Statement* pada DCL antara lain *grant*, *revoke*, *set* dan *lock table*. (Wongso, 2015).

2.5 MySQL

MySQL (*My Struktur Query Language*) adalah bahasa yang digunakan untuk mengelola data RDBMS. Pada MySQL dikenal istilah *database* dan tabel. MySQL adalah DBMS yang *open source* dengan dua bentuk lisensi, yaitu *Free Software* (perangkat lunak bebas) dan *shareware* (perangkat lunak berpemilik yang penggunaannya terbatas). Jadi MySQL adalah *database* server yang gratis dengan lisensi GNU *General Public License* (GPL) sehingga dapat dipakai untuk keperluan pribadi atau komersial tanpa harus membayar lisensi yang ada. (Kalsum Siregar et al., 2024).

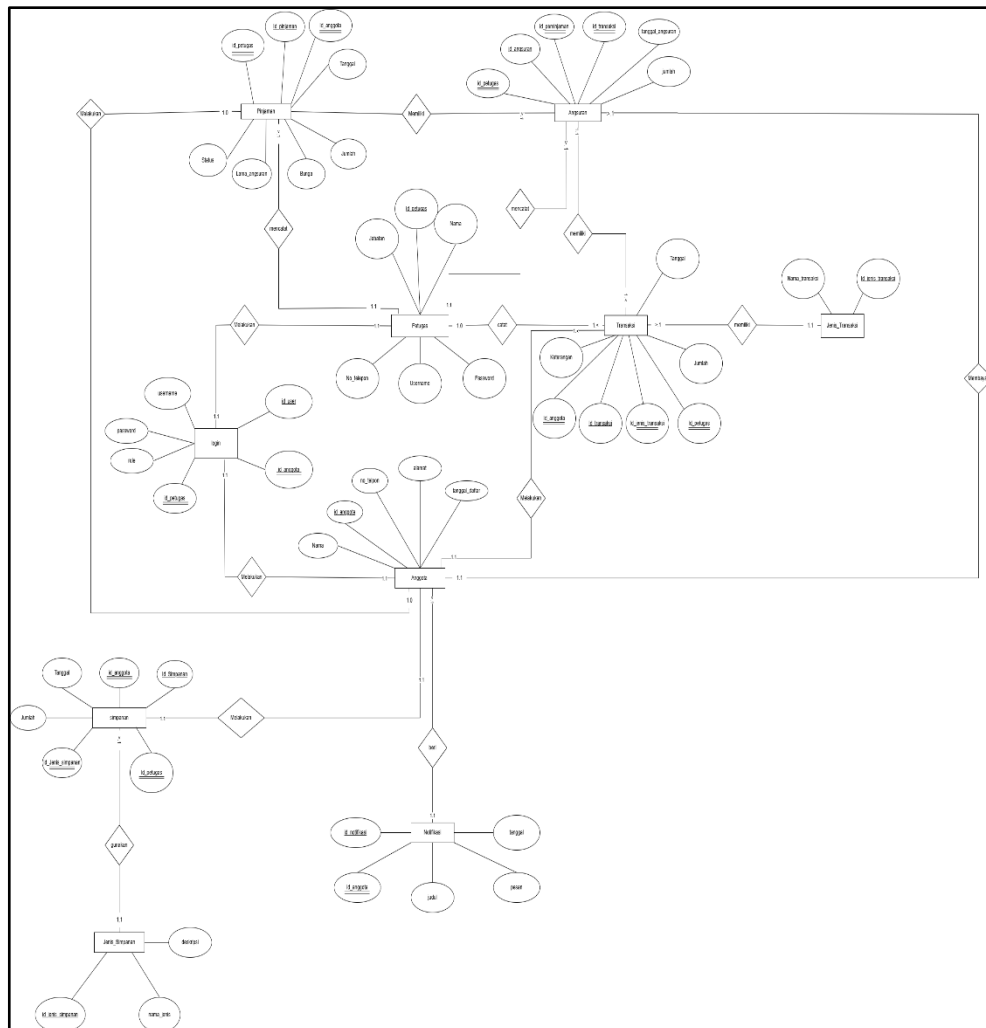
2.6 Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) adalah representasi visual dari hubungan antara entitas yang terlibat dalam sebuah sistem. Dalam ERD, entitas adalah objek nyata atau abstrak yang dapat diidentifikasi secara unik. Masing-masing entitas memiliki atribut, yang merupakan informasi yang dimiliki oleh entitas tersebut. Relasi di antara entitas menunjukkan bagaimana dua entitas atau lebih saling terkait. ERD sering kali digunakan dalam pengembangan basis data karena dapat membantu memvisualisasikan bagaimana data dalam sistem akan diatur dan diakses. Dengan menggunakan simbol-simbol tertentu, ERD memberikan pandangan yang mudah dipahami tentang struktur data, sehingga memudahkan proses analisis dan desain basis data (Sihotang et al., 2021).

Komponen-komponen pembentuk *Entity Relationship Diagram* (ERD) yaitu entitas, atribut, relasi, Relasi 1:1, relasi 1:N, relasi N:N. entitas adalah Individu yang mewakili suatu objek dan dapat dibedakan dengan objek yang lain. yang lain. Atribut adalah properti yang dimiliki oleh suatu entitas, dimana dapat mendeskripsikan karakteristik dari entitas tersebut. Relasi 1 : 1 adalah relasi yang menunjukkan bahwa setiap entitas pada himpunan entitas pertama berhubungan dengan paling banyak satu entitas pada himpunan entitas kedua. Relasi 1 : N adalah relasi yang menunjukkan bahwa hubungan antara entitas pertama dengan entitas kedua adalah satu banding banyak atau sebaliknya. Relasi N:N Hubungan ini menunjukkan bahwa setiap entitas pada himpunan entitas yang pertama dapat berhubungan dengan banyak entitas pada himpunan entitas yang kedua, demikian juga sebaliknya (Dwi Praba & Safitri, n.d.).

BAB III

HASIL DAN PEMBAHASAN



Gambar 1. ERD

Penjelasan :

ERD ini menggambarkan hubungan antar entitas dalam sistem koperasi simpan pinjam. Entitas utama adalah Anggota, yang memiliki atribut seperti id_anggota, nama, alamat, no_telepon, dan tempat_daftar. Setiap anggota dapat memiliki user login melalui entitas User, yang memiliki atribut username, password, dan role, dengan relasi satu ke satu (1:1). Selain itu, seorang anggota dapat memiliki banyak Pinjaman (1:>), yang menyimpan informasi seperti id_pinjaman, jumlah, tanggal, bunga, lama_angsuran, dan status. Pinjaman ini ditangani oleh Petugas (1:>), yang memiliki atribut id_petugas, nama, dan jabatan.

Pinjaman memiliki relasi ke entitas Angsuran (1:>), di mana setiap pinjaman dapat memiliki banyak angsuran dengan data seperti id_angsuran, jumlah, tanggal, dan

tempat_pembayaran. Angsuran juga berelasi dengan Petugas dan Anggota secara terpisah (masing-masing 1:>). Transaksi juga muncul sebagai entitas penting yang mencatat semua aktivitas keuangan (1:> dengan anggota), dan memiliki atribut seperti id_transaksi, jumlah, tanggal, dan keterangan. Setiap transaksi dikategorikan melalui entitas Jenis_Transaksi (1:>), dengan atribut id_jenis_transaksi dan nama_jenis.

Untuk sistem penyimpanan, terdapat entitas Simpanan yang berelasi dengan Anggota (1:>) dan memiliki atribut id_simpanan, jumlah, dan tanggal. Simpanan dikelompokkan berdasarkan Jenis_Simpanan (1:>), yang memiliki id_jenis_simpanan dan nama_jenis, serta dideskripsikan oleh Deskripsi (1:1 ke Jenis_Simpanan). Anggota juga bisa menerima Notifikasi, yang berisi pesan tertentu dan berelasi satu ke banyak (1:>) dari Anggota. Selain itu, proses beli barang oleh anggota dicatat dalam entitas Beli (1:>), dan barang tersebut berasal dari entitas Barang (1:>), dengan atribut id_barang, nama_barang, dan stok.

1. Membuat Database dan Tabel

Printscreen Source Code :

```
1 • create database koperasi;
2 • use koperasi;
3
4 • create table login (
5     id_user int auto_increment primary key,
6     username varchar(50) not null unique,
7     password varchar(255) not null,
8     role enum('petugas', 'anggota') not null,
9     id_petugas int,
10    id_anggota int,
11    foreign key (id_petugas) references petugas (id_petugas),
12    foreign key (id_anggota) references anggota (id_anggota)
13 );
14
15 • create table petugas (
16     id_petugas int auto_increment primary key,
17     nama varchar(100) not null,
18     jabatan varchar(50),
19     no_telepon varchar(20),
20     username varchar(50) unique,
21     password varchar(255)
22 );
23
24 • create table anggota (
25     id_anggota int auto_increment primary key,
26     nama varchar(100) not null,
27     no_telepon varchar(20),
28     alamat text,
29     tanggal_daftar date not null
```

```

29     tanggal_daftar date not null
30 };
31
32 • create table jenis_simpanan (
33     id_jenis_simpanan int auto_increment primary key,
34     nama_jenis varchar(50) not null,
35     deskripsi text
36 );
37
38 • create table simpanan (
39     id_simpanan int auto_increment primary key,
40     id_anggota int not null,
41     id_jenis_simpanan int not null,
42     id_petugas int not null,
43     tanggal date not null,
44     jumlah decimal(15,2) not null,
45     foreign key (id_anggota) references anggota (id_anggota),
46     foreign key (id_jenis_simpanan) references jenis_simpanan (id_jenis_simpanan),
47     foreign key (id_petugas) references petugas (id_petugas)
48 );
49
50 • create table jenis_transaksi (
51     id_jenis_transaksi int auto_increment primary key,
52     nama_transaksi varchar(50) not null
53 );

```

```

55 • create table transaksi (
56     id_transaksi int auto_increment primary key,
57     id_anggota int not null,
58     id_petugas int not null,
59     id_jenis_transaksi int not null,
60     tanggal date not null,
61     jumlah decimal(15,2) not null,
62     keterangan text,
63     foreign key (id_anggota) references anggota (id_anggota),
64     foreign key (id_petugas) references petugas (id_petugas),
65     foreign key (id_jenis_transaksi) references jenis_transaksi (id_jenis_transaksi)
66 );
67
68 • create table pinjaman (
69     id_pinjaman int auto_increment primary key,
70     id_anggota int not null,
71     id_petugas int not null,
72     tanggal date not null,
73     jumlah decimal(15,2) not null,
74     bunga decimal(15,2) not null,
75     lama_angsuran int not null,
76     status enum('lunas','belum lunas') not null,
77     foreign key (id_anggota) references anggota (id_anggota),
78     foreign key (id_petugas) references petugas (id_petugas)
79 );

```

```

81 • create table angsuran (
82     id_angsuran int auto_increment primary key,
83     id_pinjaman int not null,
84     id_petugas int not null,
85     id_transaksi int not null,
86     tanggal_angsuran date not null,
87     jumlah decimal(15,2) not null,
88     foreign key (id_pinjaman) references pinjaman (id_pinjaman),
89     foreign key (id_petugas) references petugas (id_petugas),
90     foreign key (id_transaksi) references transaksi (id_transaksi)
91 );
92
93 • create table notifikasi (
94     id_notifikasi int auto_increment primary key,
95     id_anggota int not null,
96     judul varchar(100),
97     pesan text,
98     tanggal date,
99     foreign key (id_anggota) references anggota (id_anggota)
100 );
101

```

Gambar 2. Printscreen Source Code membuat Database dan Tabel

Source Code :

create database koperasi;

use koperasi;

```

create table login (
id_user int auto_increment primary key,
username varchar(50) not null unique,
password varchar(255) not null,
role enum('petugas','anggota') not null,
id_petugas int,
id_anggota int,
foreign key (id_petugas) references petugas (id_petugas),
foreign key (id_anggota) references anggota (id_anggota)
);

create table petugas (
id_petugas int auto_increment primary key,
nama varchar(100) not null,
jabatan varchar(50),
no_telepon varchar(20),
username varchar(50) unique,
password varchar(255)
);

create table anggota (
id_anggota int auto_increment primary key,
nama varchar(100) not null,
no_telepon varchar(20),
alamat text,
tanggal_daftar date not null
);

create table jenis_simpanan (
id_jenis_simpanan int auto_increment primary key,
nama_jenis varchar(50) not null,
deskripsi text
);

create table simpanan (
id_simpanan int auto_increment primary key,

```

```

id_anggota int not null,
id_jenis_simpanan int not null,
id_petugas int not null,
tanggal date not null,
jumlah decimal(15,2) not null,
foreign key (id_anggota) references anggota (id_anggota),
foreign key (id_jenis_simpanan) references jenis_simpanan (id_jenis_simpanan),
foreign key (id_petugas) references petugas (id_petugas)
);
create table jenis_transaksi (
id_jenis_transaksi int auto_increment primary key,
nama_transaksi varchar(50) not null
);
create table transaksi (
id_transaksi int auto_increment primary key,
id_anggota int not null,
id_petugas int not null,
id_jenis_transaksi int not null,
tanggal date not null,
jumlah decimal(15,2) not null,
keterangan text,
foreign key (id_anggota) references anggota (id_anggota),
foreign key (id_petugas) references petugas (id_petugas),
foreign key (id_jenis_transaksi) references jenis_transaksi (id_jenis_transaksi)
);
create table pinjaman (
id_pinjaman int auto_increment primary key,
id_anggota int not null,
id_petugas int not null,
tanggal date not null,
jumlah decimal(15,2) not null,
bunga decimal(15,2) not null,

```

```

lama_angsuran int not null,
status enum('lunas','belum lunas') not null,
foreign key (id_anggota) references anggota (id_anggota),
foreign key (id_petugas) references petugas (id_petugas)
);

create table angsuran (
id_angsuran int auto_increment primary key,
id_pinjaman int not null,
id_petugas int not null,
id_transaksi int not null,
tanggal_angsuran date not null,
jumlah decimal(15,2) not null,
foreign key (id_pinjaman) references pinjaman (id_pinjaman),
foreign key (id_petugas) references petugas (id_petugas),
foreign key (id_transaksi) references transaksi (id_transaksi)
);

create table notifikasi (
id_notifikasi int auto_increment primary key,
id_anggota int not null,
judul varchar(100),
pesan text,
tanggal date,
foreign key (id_anggota) references anggota (id_anggota)
);

```

Penjelasan Source Code :

Pertama, basis data bernama koperasi dibuat dan diaktifkan menggunakan perintah **CREATE DATABASE** dan **USE**. Kemudian, dibuat beberapa tabel dengan hubungan relasional antar entitas yang mencerminkan sistem koperasi tersebut. Tabel login menyimpan data autentikasi pengguna dengan atribut seperti username, password, role, dan relasi ke tabel petugas maupun anggota melalui id_petugas dan id_anggota. Kolom role dibatasi nilainya hanya boleh 'petugas' atau 'anggota', menggunakan tipe ENUM.

Tabel petugas menyimpan informasi pegawai koperasi, termasuk nama, jabatan, no_telepon, serta kredensial seperti username dan password. Tabel ini terhubung dengan banyak entitas lain karena petugas bertindak sebagai pengelola dalam banyak transaksi. Tabel anggota mencatat data pribadi anggota koperasi seperti nama, alamat, no_telepon, dan tanggal_daftar. Anggota adalah entitas utama yang bisa melakukan simpanan, pinjaman, menerima notifikasi, dan melakukan transaksi.

Tabel jenis_simpanan menyimpan jenis-jenis simpanan seperti simpanan wajib, pokok, dan sukarela. Setiap jenis memiliki deskripsi yang menjelaskan kegunaannya. Tabel simpanan mencatat setiap transaksi penyimpanan uang oleh anggota, mencakup id_anggota, id_jenis_simpanan, id_petugas, tanggal, dan jumlah. Tabel ini menghubungkan tiga entitas: anggota, jenis_simpanan, dan petugas.

Tabel jenis_transaksi menyimpan tipe transaksi seperti pembayaran angsuran, pembelian barang, dan lainnya, dengan hanya dua kolom: id_jenis_transaksi dan nama_transaksi. Tabel transaksi mencatat semua aktivitas keuangan anggota. Setiap transaksi berkaitan dengan seorang anggota, seorang petugas, dan jenis transaksi tertentu. Informasi detail seperti tanggal, jumlah, dan keterangan juga dicatat.

Tabel pinjaman merekam pinjaman uang yang dilakukan anggota. Setiap pinjaman mencakup jumlah, bunga, lama_angsuran, status, dan relasi ke anggota dan petugas. Tabel angsuran menyimpan data cicilan atas pinjaman. Angsuran memiliki relasi ke pinjaman, petugas, dan transaksi, dan mencatat tanggal_angsuran serta jumlah. Terakhir, tabel notifikasi digunakan untuk mengirim pesan atau pengingat ke anggota, dengan mencatat judul, pesan, tanggal, dan relasi ke id_anggota.

Printscreen Output :



Gambar 3. Printscreen Output Membuat Database dan Tabel

Penjelasan Output :

Kode SQL tersebut menghasilkan struktur database bernama koperasi yang terdiri dari beberapa tabel utama seperti login, petugas, anggota, simpanan, transaksi, pinjaman, angsuran, dan notifikasi. Setiap tabel memiliki atribut spesifik dan saling terhubung melalui relasi foreign key untuk memastikan integritas data. Tabel login mengelola autentikasi pengguna berdasarkan peran (petugas atau anggota), sementara tabel petugas dan anggota menyimpan data personal masing-masing. Tabel simpanan dan jenis_simpanan mencatat aktivitas penyimpanan dana, sedangkan pinjaman dan angsuran mengelola data pinjaman dan pelunasannya. Aktivitas keuangan umum dicatat di tabel transaksi yang terhubung ke jenis_transaksi, dan pemberitahuan ke anggota disimpan dalam tabel notifikasi. Struktur ini membentuk sistem informasi koperasi yang saling terintegrasi dan siap digunakan.

2. Menginput data

Printscreen Source Code :

```
102 • insert into petugas (nama, jabatan, no_telepon, username, password) values
103 ('Dian Prasetyo', 'Manajer', '081234567890', 'dian', 'pass123'),
104 ('Rina Marlina', 'Staff Simpanan', '081298765432', 'rina', 'pass456'),
105 ('Budi Santoso', 'Kasir', '082112345678', 'budi', 'pass789'),
106 ('Eka Widya', 'Admin', '083112233445', 'eka', 'pass321'),
107 ('Farhan Yusuf', 'Staff Pinjaman', '084433221100', 'farhan', 'pass654');
108
109 • insert into anggota (nama, no_telepon, alamat, tanggal_daftar) values
110 ('Ahmad Fauzi', '082233445566', 'Jl. Merdeka No. 10', '2025-01-10'),
111 ('Siti Aminah', '081245678900', 'Jl. Kenanga No. 15', '2025-02-15'),
112 ('Rudi Hartono', '085566778899', 'Jl. Melati No. 3', '2025-03-05'),
113 ('Lilis Suryani', '086677889900', 'Jl. Anggrek No. 8', '2025-04-01'),
114 ('Teguh Wibowo', '087788990011', 'Jl. Mawar No. 12', '2025-04-10'),
115 ('Bayu Pramana', '089912345678', 'Jl. Cendana No. 5', '2025-05-05'); -- TANPA simpanan
116
117 • insert into login (username, password, role, id_petugas, id_anggota) values
118 ('dian', 'pass123', 'petugas', 1, null),
119 ('rina', 'pass456', 'petugas', 2, null),
120 ('ahmad', '1234', 'anggota', null, 1),
121 ('siti', '5678', 'anggota', null, 2),
122 ('rudi', 'abcd', 'anggota', null, 3);
123
124 • insert into jenis_simpanan (nama_jenis, deskripsi) values
125 ('Simpanan Pokok', 'Simpanan wajib setiap anggota saat mendaftar'),
126 ('Simpanan Wajib', 'Simpanan bulanan wajib anggota'),
127 ('Simpanan Sukarela', 'Simpanan bebas sesuai kemampuan anggota'),
128 ('Simpanan Pendidikan', 'Simpanan khusus untuk biaya pendidikan'),
129 ('Simpanan Hari Raya', 'Disimpan untuk keperluan hari besar keagamaan');
130
131 • insert into simpanan (id_anggota, id_jenis_simpanan, id_petugas, tanggal, jumlah) values
132 (1, 1, 2, '2025-03-01', 100000.00),
133 (2, 2, 3, '2025-03-10', 150000.00),
134 (3, 3, 2, '2025-03-15', 50000.00),
135 (4, 1, 1, '2025-04-01', 100000.00),
136 (5, 2, 5, '2025-04-05', 150000.00);
137
138 • insert into jenis_transaksi (nama_transaksi) values
139 ('Pembayaran Angsuran'),
140 ('Penarikan Simpanan'),
141 ('Pembayaran Pinjaman'),
142 ('Setoran Simpanan'),
143 ('Pembayaran Denda');
144
```

```

145 • insert into transaksi (id_anggota, id_petugas, id_jenis_transaksi, tanggal, jumlah, keterangan) values
146 (1, 2, 1, '2025-04-10', 50000.00, 'Angsuran ke-1'),
147 (2, 3, 2, '2025-04-15', 30000.00, 'Penarikan sukarela'),
148 (3, 4, 4, '2025-04-20', 70000.00, 'Setoran bulanan'),
149 (4, 1, 3, '2025-04-25', 100000.00, 'Pembayaran pertama'),
150 (5, 5, 5, '2025-05-01', 10000.00, 'Denda keterlambatan');
151
152 • insert into pinjaman (id_anggota, id_petugas, tanggal, jumlah, bunga, lama_angsuran, status) values
153 (1, 1, '2025-03-15', 1000000.00, 100000.00, 10, 'belum lunas'),
154 (2, 2, '2025-03-20', 2000000.00, 200000.00, 12, 'belum lunas'),
155 (3, 3, '2025-04-01', 1500000.00, 150000.00, 6, 'lunas'),
156 (4, 4, '2025-04-05', 1200000.00, 120000.00, 8, 'belum lunas'),
157 (5, 5, '2025-04-10', 1800000.00, 180000.00, 9, 'belum lunas');
158
159 • insert into angsuran (id_pinjaman, id_petugas, id_transaksi, tanggal_angsuran, jumlah) values
160 (1, 2, 1, '2025-04-10', 50000.00),
161 (2, 3, 2, '2025-04-15', 100000.00),
162 (3, 4, 3, '2025-04-20', 150000.00),
163 (4, 1, 4, '2025-04-25', 120000.00),
164 (5, 5, 5, '2025-05-01', 180000.00);
165

```

```

insert into notifikasi (id_anggota, judul, pesan, tanggal) values
(1, 'Jatuh Tempo Angsuran', 'Angsuran ke-2 jatuh tempo 10 Mei', '2025-05-01'),
(2, 'Saldo Simpanan Bertambah', 'Penambahan simpanan Rp150.000', '2025-04-10'),
(3, 'Pinjaman Disetujui', 'Pinjaman Anda telah disetujui', '2025-04-01'),
(4, 'Denda Keterlambatan', 'Anda terkena denda Rp10.000', '2025-05-01'),
(5, 'Pembayaran Lunas', 'Pinjaman Anda telah lunas', '2025-05-02');

```

Gambar 4. Printscreen Source Code membuat Database dan Tabel

Source Code :

```

insert into petugas (nama, jabatan, no_telepon, username, password) values
('Dian Prasetyo', 'Manajer', '081234567890', 'dian', 'pass123'),
('Rina Marlina', 'Staff Simpanan', '081298765432', 'rina', 'pass456'),
('Budi Santoso', 'Kasir', '082112345678', 'budi', 'pass789'),
('Eka Widya', 'Admin', '083112233445', 'eka', 'pass321'),
('Farhan Yusuf', 'Staff Pinjaman', '084433221100', 'farhan', 'pass654');
insert into anggota (nama, no_telepon, alamat, tanggal_daftar) values
('Ahmad Fauzi', '082233445566', 'Jl. Merdeka No. 10', '2025-01-10'),
('Siti Aminah', '081245678900', 'Jl. Kenanga No. 15', '2025-02-15'),
('Rudi Hartono', '085566778899', 'Jl. Melati No. 3', '2025-03-05'),
('Lilis Suryani', '086677889900', 'Jl. Anggrek No. 8', '2025-04-01'),
('Teguh Wibowo', '087788990011', 'Jl. Mawar No. 12', '2025-04-10'),
('Bayu Pramana', '089912345678', 'Jl. Cendana No. 5', '2025-05-05'); -- TANPA
simpanan
insert into login (username, password, role, id_petugas, id_anggota) values
('dian', 'pass123', 'petugas', 1, null),
('rina', 'pass456', 'petugas', 2, null),
('ahmad', '1234', 'anggota', null, 1),

```

```

('siti', '5678', 'anggota', null, 2),
('rudi', 'abcd', 'anggota', null, 3);
insert into jenis_simpanan (nama_jenis, deskripsi) values
('Simpanan Pokok', 'Simpanan wajib setiap anggota saat mendaftar'),
('Simpanan Wajib', 'Simpanan bulanan wajib anggota'),
('Simpanan Sukarela', 'Simpanan bebas sesuai kemampuan anggota'),
('Simpanan Pendidikan', 'Simpanan khusus untuk biaya pendidikan'),
('Simpanan Hari Raya', 'Disimpan untuk keperluan hari besar keagamaan');
insert into simpanan (id_anggota, id_jenis_simpanan, id_petugas, tanggal,
    jumlah) values
(1, 1, 2, '2025-03-01', 100000.00),
(2, 2, 3, '2025-03-10', 150000.00),
(3, 3, 2, '2025-03-15', 50000.00),
(4, 1, 1, '2025-04-01', 100000.00),
(5, 2, 5, '2025-04-05', 150000.00);
insert into jenis_transaksi (nama_transaksi) values
('Pembayaran Angsuran'),
('Penarikan Simpanan'),
('Pembayaran Pinjaman'),
('Setoran Simpanan'),
('Pembayaran Denda');
insert into transaksi (id_anggota, id_petugas, id_jenis_transaksi, tanggal, jumlah,
    keterangan) values
(1, 2, 1, '2025-04-10', 50000.00, 'Angsuran ke-1'),
(2, 3, 2, '2025-04-15', 30000.00, 'Penarikan sukarela'),
(3, 4, 4, '2025-04-20', 70000.00, 'Setoran bulanan'),
(4, 1, 3, '2025-04-25', 100000.00, 'Pembayaran pertama'),
(5, 5, 5, '2025-05-01', 10000.00, 'Denda keterlambatan');
insert into pinjaman (id_anggota, id_petugas, tanggal, jumlah, bunga,
    lama_angsuran, status) values
(1, 1, '2025-03-15', 1000000.00, 100000.00, 10, 'belum lunas'),
(2, 2, '2025-03-20', 2000000.00, 200000.00, 12, 'belum lunas'),

```

```

(3, 3, '2025-04-01', 1500000.00, 150000.00, 6, 'lunas'),
(4, 4, '2025-04-05', 1200000.00, 120000.00, 8, 'belum lunas'),
(5, 5, '2025-04-10', 1800000.00, 180000.00, 9, 'belum lunas');
insert into angsuran (id_pinjaman, id_petugas, id_transaksi, tanggal_angsuran,
jumlah) values
(1, 2, 1, '2025-04-10', 50000.00),
(2, 3, 2, '2025-04-15', 100000.00),
(3, 4, 3, '2025-04-20', 150000.00),
(4, 1, 4, '2025-04-25', 120000.00),
(5, 5, 5, '2025-05-01', 180000.00);
insert into notifikasi (id_anggota, judul, pesan, tanggal) values
(1, 'Jatuh Tempo Angsuran', 'Angsuran ke-2 jatuh tempo 10 Mei', '2025-05-01'),
(2, 'Saldo Simpanan Bertambah', 'Penambahan simpanan Rp150.000', '2025-04-10'),
(3, 'Pinjaman Disetujui', 'Pinjaman Anda telah disetujui', '2025-04-01'),
(4, 'Denda Keterlambatan', 'Anda terkena denda Rp10.000', '2025-05-01'),
(5, 'Pembayaran Lunas', 'Pinjaman Anda telah lunas', '2025-05-02');

```

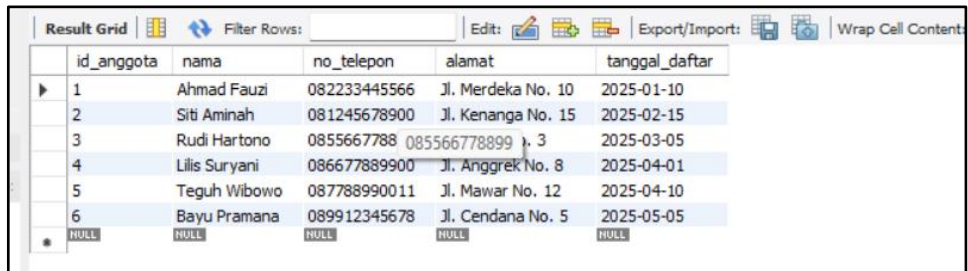
Penjelasan Source Code :

Source code SQL tersebut berfungsi untuk mengisi data awal (seeding) ke dalam sistem database koperasi. Pertama, tabel petugas diisi dengan lima data petugas lengkap dengan nama, jabatan, nomor telepon, username, dan password. Kemudian, tabel anggota diisi dengan enam anggota koperasi beserta nomor telepon, alamat, dan tanggal pendaftaran. Data login ditambahkan ke tabel login, yang menghubungkan setiap pengguna dengan perannya sebagai petugas atau anggota, serta ID terkait. Tabel jenis_simpanan berisi lima jenis simpanan berbeda, dan tabel simpanan mencatat transaksi simpanan yang dilakukan lima anggota dengan petugas masing-masing. Tabel jenis_transaksi mencantumkan lima jenis transaksi umum dalam koperasi, lalu tabel transaksi memuat data transaksi keuangan yang dilakukan anggota, termasuk tanggal, jumlah, dan keterangan.

Selanjutnya, tabel pinjaman mencatat data pinjaman yang diajukan oleh anggota, termasuk nilai pinjaman, bunga, lama angsuran, dan status pelunasan. Data pembayaran angsuran dimasukkan ke tabel angsuran, yang menghubungkan

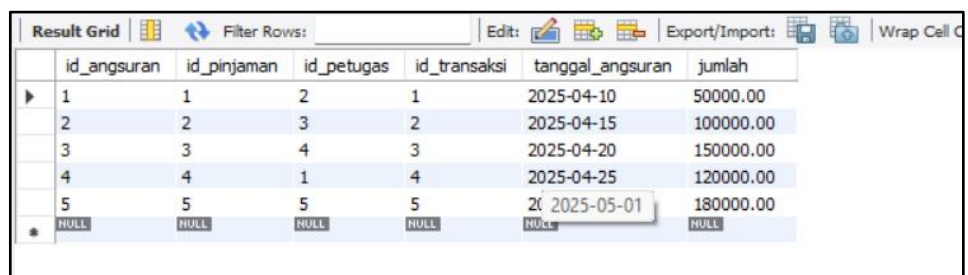
pinjaman, petugas, dan transaksi terkait. Terakhir, tabel notifikasi diisi dengan lima pesan yang dikirim ke anggota sebagai informasi penting mengenai angsuran, simpanan, pinjaman, dan denda. Keseluruhan script ini menciptakan contoh data yang lengkap dan realistis untuk mendukung pengujian dan pengoperasian sistem manajemen koperasi.

Printscreen Output :



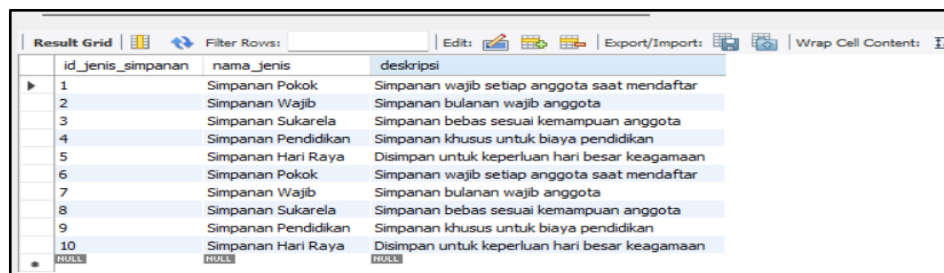
	id_anggota	nama	no_telepon	alamat	tanggal_daftar
1	1	Ahmad Fauzi	082233445566	Jl. Merdeka No. 10	2025-01-10
2	2	Siti Aminah	081245678900	Jl. Kenanga No. 15	2025-02-15
3	3	Rudi Hartono	0855667788	085566778899 No. 3	2025-03-05
4	4	Lilis Suryani	086677889900	Jl. Anggrek No. 8	2025-04-01
5	5	Teguh Wibowo	087788990011	Jl. Mawar No. 12	2025-04-10
6	6	Bayu Pramana	089912345678	Jl. Cendana No. 5	2025-05-05
*	NULL	NULL	NULL	NULL	NULL

Gambar 5. Printscreen Output Memasukkan data ke tabel anggota



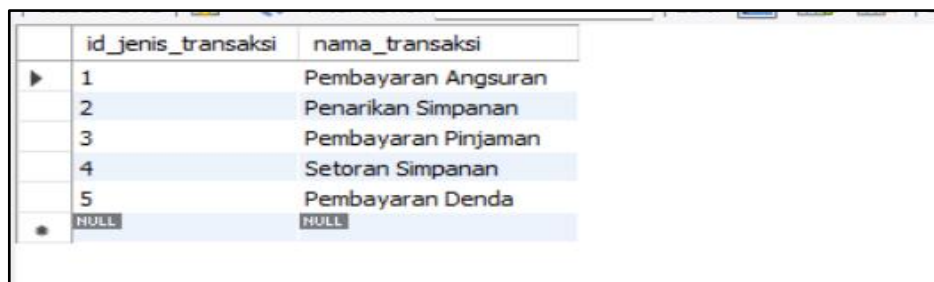
	id_angsuran	id_pinjaman	id_petugas	id_transaksi	tanggal_angsuran	jumlah
1	1	1	2	1	2025-04-10	50000.00
2	2	2	3	2	2025-04-15	100000.00
3	3	3	4	3	2025-04-20	150000.00
4	4	4	1	4	2025-04-25	120000.00
5	5	5	5	5	2025-05-01	180000.00
*	NULL	NULL	NULL	NULL	NULL	NULL

Gambar 6. Printscreen Output Memasukkan data ke tabel angsuran



	id_jenis_simpanan	nama_jenis	deskripsi
1	1	Simpanan Pokok	Simpanan wajib setiap anggota saat mendaftar
2	2	Simpanan Wajib	Simpanan bulanan wajib anggota
3	3	Simpanan Sukarela	Simpanan bebas sesuai kemampuan anggota
4	4	Simpanan Pendidikan	Simpanan khusus untuk biaya pendidikan
5	5	Simpanan Hari Raya	Disimpan untuk keperluan hari besar keagamaan
6	6	Simpanan Pokok	Simpanan wajib setiap anggota saat mendaftar
7	7	Simpanan Wajib	Simpanan bulanan wajib anggota
8	8	Simpanan Sukarela	Simpanan bebas sesuai kemampuan anggota
9	9	Simpanan Pendidikan	Simpanan khusus untuk biaya pendidikan
10	10	Simpanan Hari Raya	Disimpan untuk keperluan hari besar keagamaan
*	NULL	NULL	NULL

Gambar 7. Printscreen Output Memasukkan data ke tabel jenis simpanan



	id_jenis_transaksi	nama_transaksi
1	1	Pembayaran Angsuran
2	2	Penarikan Simpanan
3	3	Pembayaran Pinjaman
4	4	Setoran Simpanan
5	5	Pembayaran Denda
*	NULL	NULL

Gambar 8. Printscreen Output Memasukkan data ke tabel jenis transaksi

	id_user	username	password	role	id_petugas	id_anggota
▶	1	dian	pass123	petugas	1	NULL
	2	rina	pass456	petugas	2	NULL
	3	ahmad	1234	anggota	NULL	1
	4	siti	5678	anggota	NULL	2
	5	rudi	abcd	anggota	NULL	3
*	NULL	NULL	NULL	NULL	NULL	NULL

Gambar 9. Printscreen Output Memasukkan data ke tabel login

	id_notifikasi	id_anggota	judul	pesan	tanggal
▶	1	1	Jatuh Tempo Angsuran	Angsuran ke-2 jatuh tempo 10 Mei	2025-05-01
	2	2	Saldo Simpanan Bertambah	Penambahan simpanan Rp150.000	2025-04-10
	3	3	Pin Pinjaman Disetujui	Pinjaman Anda telah disetujui	2025-04-01
	4	4	Denda Keterlambatan	Anda terkena denda Rp10.000	2025-05-01
	5	5	Pembayaran Lunas	Pinjaman Anda telah lunas	2025-05-02
	6	1	Jatuh Tempo Angsuran	Angsuran ke-2 jatuh tempo 10 Mei	2025-05-01
	7	2	Saldo Simpanan Bertambah	Penambahan simpanan Rp150.000	2025-04-10
	8	3	Pinjaman Disetujui	Pinjaman Anda telah disetujui	2025-04-01
	9	4	Denda Keterlambatan	Anda terkena denda Rp10.000	2025-05-01
	10	5	Pembayaran Lunas	Pinjaman Anda telah lunas	2025-05-02
*	NULL	NULL	NULL	NULL	NULL

Gambar 10. Printscreen Output Memasukkan data ke tabel notifikasi

	id_petugas	nama	jabatan	no_telepon	username	password
▶	1	Dian Prasetyo	Manajer	081234567890	dian	pass123
	2	Rina Marlina	Staff Simpanan	081298765432	rina	pass456
	3	Budi Santoso	Kasir	082112345678	budi	pass789
	4	Eka Widya	Admin	083112233445	eka	pass321
	5	Farhan Yusuf	Staff Pinjaman	084433221100	farhan	pass654
*	NULL	NULL	NULL	NULL	NULL	NULL

Gambar 11. Printscreen Output Memasukkan data ke tabel Petugas

	id_pinjaman	id_anggota	id_petugas	tanggal	jumlah	bunga	lama_angsuran	status
▶	1	1	1	2025-03-15	1000000.00	100000.00	10	belum lunas
	2	2	2	2025-03-20	2000000.00	200000.00	12	belum lunas
	3	3	3	2025-04-01	1500000.00	150000.00	6	lunas
	4	4	4	2025-04-05	1200000.00	120000.00	8	belum lunas
	5	5	5	2025-04-10	1800000.00	180000.00	9	belum lunas
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Gambar 12. Printscreen Output Memasukkan data ke tabel Pinjaman

	id_simpanan	id_anggota	id_jenis_simpanan	id_petugas	tanggal	jumlah
▶	11	1	1	2	2025-03-01	100000.00
	12	2	2	3	2025-03-10	150000.00
	13	3	3	2	2025-03-15	50000.00
	14	4	1	1	2025-04-01	100000.00
	15	5	2	5	2025-04-05	150000.00
*	NULL	NULL	NULL	NULL	NULL	NULL

Gambar 13. Printscreen Output Memasukkan data ke tabel Simpanan

	id_transaksi	id_anggota	id_petugas	id_jenis_transaksi	tanggal	jumlah	keterangan
▶	1	1	2	1	2025-04-10	50000.00	Angsuran ke-1
	2	2	3	2	2025-04-15	30000.00	Penarikan sukarela
	3	3	4	4	2025-04-20	70000.00	Setoran bulanan
	4	4	1	3	2025-04-25	100000.00	Pembayaran pertama
	5	5	5	5	2025-05-01	10000.00	Denda keterlambatan
	6	1	2	1	2025-04-10	50000.00	Angsuran ke-1
	7	2	3	2	2025-04-15	30000.00	Penarikan sukarela
	8	3	4	4	2025-04-20	70000.00	Setoran bulanan
	9	4	1	3	2025-04-25	100000.00	Pembayaran pertama
	10	5	5	5	2025-05-01	10000.00	Denda keterlambatan
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Gambar 14. Printscreen Output Memasukkan data ke tabel Transaksi

Penjelasan Output :

Output dari kode SQL tersebut adalah data awal (dummy data) yang berhasil dimasukkan ke dalam berbagai tabel di basis data sistem koperasi. Setelah seluruh perintah **INSERT INTO** dijalankan, masing-masing tabel akan terisi dengan informasi yang diperlukan untuk menjalankan sistem, seperti daftar petugas dan anggota, jenis-jenis simpanan, riwayat simpanan, transaksi keuangan, pinjaman serta angsurannya, dan notifikasi kepada anggota. Data ini memungkinkan aplikasi koperasi untuk langsung menampilkan informasi dan melakukan operasi seperti login, mencatat transaksi, dan menampilkan laporan pinjaman dan simpanan berdasarkan data yang sudah tersedia.

3. Join

Printscreen Source Code :

```

173 • select
174     a.id_anggota,
175     a.nama as nama_anggota,
176     s.id_simpanan,
177     s.jumlah
178 from anggota a
179 left join simpanan s on a.id_anggota = s.id_anggota;
180
181 • select
182     a.id_anggota,
183     a.nama as nama_anggota,
184     s.id_simpanan,
185     s.jumlah
186 from anggota a
187 right join simpanan s on a.id_anggota = s.id_anggota;
188
189 • select
190     a.id_anggota,
191     a.nama as nama_anggota,
192     s.id_simpanan,
193     s.jumlah,
194     s.tanggal
195 from anggota a
196 inner join simpanan s on a.id_anggota = s.id_anggota;

```



```

198 • select
199     a.id_anggota,
200     a.nama as nama_anggota,
201     s.id_simpanan,
202     s.jumlah,
203     s.tanggal
204 from anggota a
205 left join simpanan s on a.id_anggota = s.id_anggota
206 union
207 select
208     a.id_anggota,
209     a.nama as nama_anggota,
210     s.id_simpanan,
211     s.jumlah,
212     s.tanggal
213 from anggota a
214 right join simpanan s on a.id_anggota = s.id_anggota;
215

```

Gambar 15. Printscreen Source Code membuat Database dan Tabel

Source Code :

```

select
    a.id_anggota,
    a.nama as nama_anggota,
    s.id_simpanan,
    s.jumlah
from anggota a
left join simpanan s on a.id_anggota = s.id_anggota;

select
    a.id_anggota,
    a.nama as nama_anggota,
    s.id_simpanan,
    s.jumlah
from anggota a
right join simpanan s on a.id_anggota = s.id_anggota;

select
    a.id_anggota,
    a.nama as nama_anggota,
    s.id_simpanan,
    s.jumlah,
    s.tanggal
from anggota a
inner join simpanan s on a.id_anggota = s.id_anggota;

select

```

```

a.id_anggota,
a.nama as nama_anggota,
s.id_simpanan,
s.jumlah,
s.tanggal
from anggota a
left join simpanan s on a.id_anggota = s.id_anggota
union
select
a.id_anggota,
a.nama as nama_anggota,
s.id_simpanan,
s.jumlah,
s.tanggal
from anggota a
right join simpanan s on a.id_anggota = s.id_anggota;

```

Penjelasan Source Code :

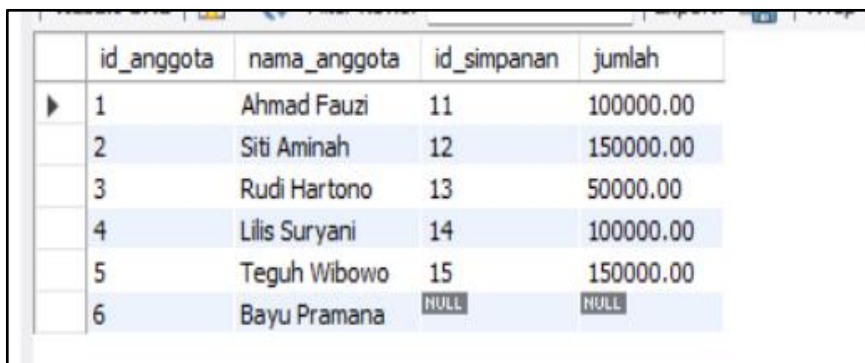
Query pertama menggunakan LEFT JOIN yang berarti akan menampilkan seluruh data dari tabel anggota, dan jika ada data yang cocok di tabel simpanan (berdasarkan id_anggota), maka data tersebut akan ikut ditampilkan. Jika tidak ada simpanan, maka kolom dari tabel simpanan akan bernilai NULL. Ini berguna untuk mencari anggota yang sudah atau belum memiliki simpanan.

Query kedua menggunakan RIGHT JOIN yang berarti akan menampilkan seluruh data dari tabel simpanan, dan jika ada kecocokan dengan anggota (berdasarkan id_anggota), maka data anggota akan ikut ditampilkan. Jika tidak ada data anggota yang cocok, maka kolom dari anggota akan bernilai NULL. Ini berguna untuk mengetahui apakah ada data simpanan yang tidak terhubung dengan data anggota, misalnya karena kesalahan entri data.

Query ketiga menggunakan INNER JOIN, yang berarti hanya akan menampilkan data yang memiliki kecocokan di kedua tabel, yaitu anggota yang memang memiliki simpanan. Query ini tidak akan menampilkan anggota yang tidak memiliki simpanan, atau simpanan yang tidak memiliki anggota yang terdaftar.

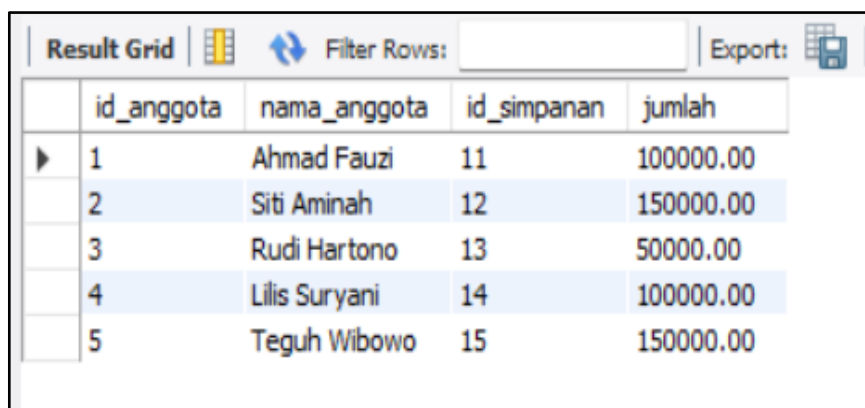
Query keempat adalah gabungan **LEFT JOIN** dan **RIGHT JOIN** yang digabung menggunakan **UNION**. Tujuannya adalah untuk mensimulasikan **FULL OUTER JOIN**, yaitu menggabungkan semua data dari kedua tabel, baik yang memiliki pasangan maupun tidak. Jika sebuah anggota tidak memiliki simpanan, atau sebuah simpanan tidak memiliki anggota yang cocok, semuanya tetap ditampilkan. Hal ini berguna untuk memastikan tidak ada data yang terlewat, dan bisa dimanfaatkan untuk audit atau pengecekan kelengkapan data.

Prinstscreen Output :



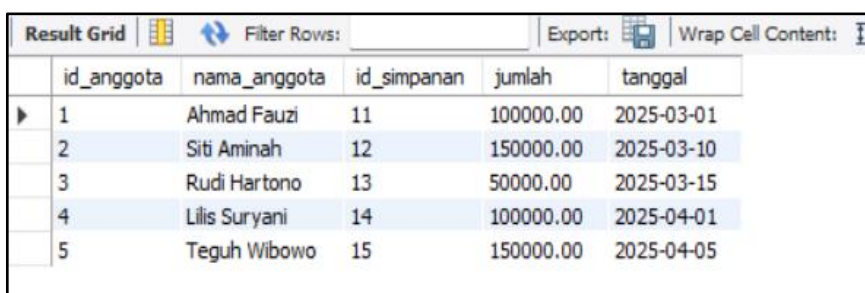
	id_anggota	nama_anggota	id_simpanan	jumlah
▶	1	Ahmad Fauzi	11	100000.00
	2	Siti Aminah	12	150000.00
	3	Rudi Hartono	13	50000.00
	4	Lilis Suryani	14	100000.00
	5	Teguh Wibowo	15	150000.00
	6	Bayu Pramana	NULL	NULL

Gambar 16. Printscreen Output Left join



	id_anggota	nama_anggota	id_simpanan	jumlah
▶	1	Ahmad Fauzi	11	100000.00
	2	Siti Aminah	12	150000.00
	3	Rudi Hartono	13	50000.00
	4	Lilis Suryani	14	100000.00
	5	Teguh Wibowo	15	150000.00

Gambar 17. Printscreen Output Right join



	id_anggota	nama_anggota	id_simpanan	jumlah	tanggal
▶	1	Ahmad Fauzi	11	100000.00	2025-03-01
	2	Siti Aminah	12	150000.00	2025-03-10
	3	Rudi Hartono	13	50000.00	2025-03-15
	4	Lilis Suryani	14	100000.00	2025-04-01
	5	Teguh Wibowo	15	150000.00	2025-04-05

Gambar 18. Printscreen Output Membuat Inner join

Result Grid					
Filter Rows:					
Export: Wrap Cell Content:					
	id_anggota	nama_anggota	id_simpanan	jumlah	tanggal
▶	1	Ahmad Fauzi	11	100000.00	2025-03-01
	2	Siti Aminah	12	150000.00	2025-03-10
	3	Rudi Hartono	13	50000.00	2025-03-15
	4	Lilis Suryani	14	100000.00	2025-04-01
	5	Teguh Wibowo	15	150000.00	2025-04-05
	6	Bayu Pramana	NULL	NULL	NULL

Gambar 19. Printscreen Output Membuat Full join

Penjelasan Output :

Output dari keempat query SQL tersebut adalah hasil gabungan informasi antara data anggota dan data simpanan, namun dengan variasi jenis *join* yang memengaruhi baris mana yang akan ditampilkan. Query pertama menghasilkan semua anggota, termasuk yang belum memiliki simpanan; anggota yang tidak memiliki simpanan akan menampilkan nilai NULL di kolom simpanan. Query kedua menampilkan semua data simpanan, bahkan jika tidak ada anggota yang cocok—meskipun dalam data yang diberikan, setiap simpanan memiliki anggota, sehingga hasilnya mirip dengan INNER JOIN. Query ketiga hanya menampilkan anggota yang benar-benar memiliki simpanan, sehingga anggota yang belum pernah menyimpan tidak akan muncul. Sementara itu, query keempat menggabungkan hasil LEFT JOIN dan RIGHT JOIN menggunakan UNION, sehingga output-nya mencakup semua kombinasi yang mungkin dari anggota dan simpanan—baik yang memiliki pasangan maupun yang tidak—memberikan gambaran menyeluruh dari seluruh data yang tersedia.

4. Alias dan Operator

Printscreen Source Code :

```
-- Alias --
SELECT
    a.nama AS nama_anggota,
    s.jumlah AS jumlah_simpanan,
    js.nama_jenis AS jenis_simpanan
FROM simpanan s
JOIN anggota a ON s.id_anggota = a.id_anggota
JOIN jenis_simpanan js ON s.id_jenis_simpanan = js.id_jenis_simpanan;
```

Gambar 20. Printscreen Source Code Alias

```

SELECT
    a.nama AS nama_anggota,
    p.jumlah + p.bunga AS total_pinjaman,
    p.jumlah - p.bunga AS pokok_pinjaman
FROM pinjaman p
JOIN anggota a ON p.id_anggota = a.id_anggota
WHERE p.status = 'belum lunas';

```

Gambar 21. Printscreen Source Code membuat Operator

Source Code :

```

SELECT
    a.nama AS nama_anggota,
    s.jumlah AS jumlah_simpanan,
    js.nama_jenis AS jenis_simpanan
FROM simpanan s
JOIN anggota a ON s.id_anggota = a.id_anggota
JOIN jenis_simpanan js ON s.id_jenis_simpanan = js.id_jenis_simpanan;

```

-- Operator

```

SELECT
    a.nama AS nama_anggota,
    p.jumlah + p.bunga AS total_pinjaman,
    p.jumlah - p.bunga AS pokok_pinjaman
FROM pinjaman p
JOIN anggota a ON p.id_anggota = a.id_anggota
WHERE p.status = 'belum lunas';

```

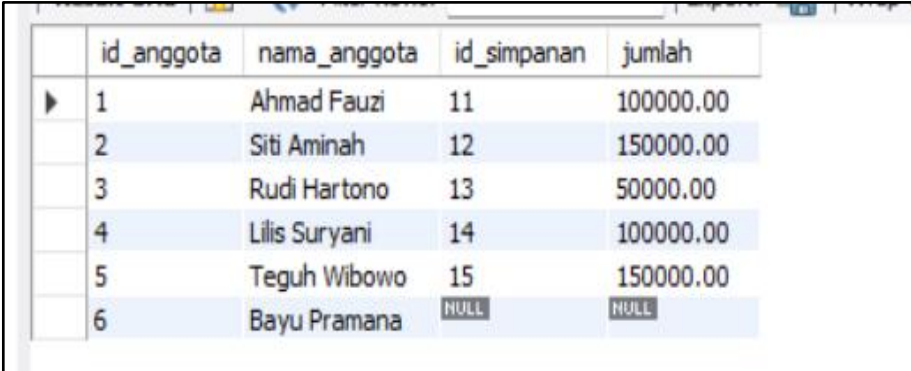
Penjelasan Source Code :

Query pertama digunakan untuk menampilkan data simpanan anggota koperasi dengan menggabungkan tiga tabel: simpanan, anggota, dan jenis_simpanan. Tabel simpanan di-join dengan tabel anggota berdasarkan id_anggota agar dapat menampilkan nama dari anggota yang melakukan simpanan. Selanjutnya, tabel simpanan juga di-join dengan tabel jenis_simpanan berdasarkan id_jenis_simpanan untuk menampilkan jenis simpanan yang dilakukan, seperti "simpanan wajib" atau

"simpanan pokok". Kolom hasil yang ditampilkan adalah nama_anggota, jumlah_simpanan, dan jenis_simpanan, yang masing-masing mewakili nama anggota, nominal simpanan, serta jenisnya.

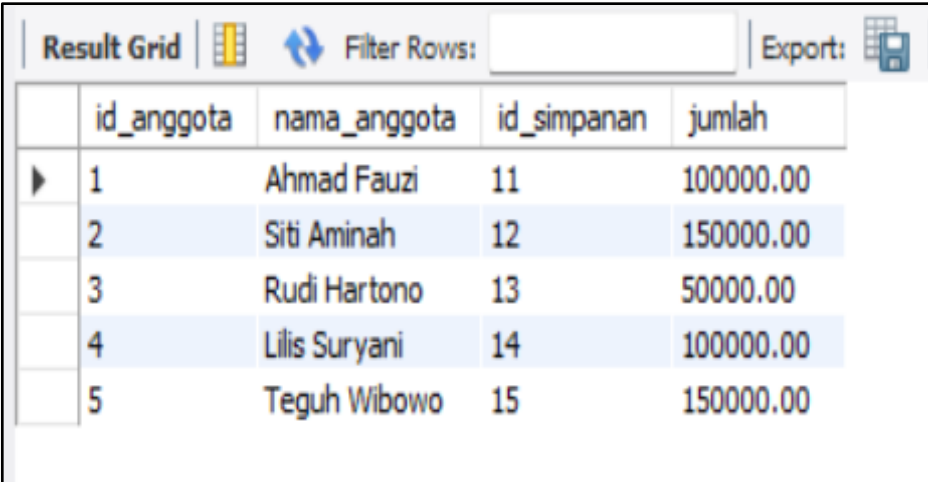
Query kedua digunakan untuk menampilkan data pinjaman anggota yang statusnya masih *belum lunas*. Tabel pinjaman digabungkan dengan tabel anggota berdasarkan id_anggota, sehingga informasi nama peminjam bisa ditampilkan. Query ini juga memanfaatkan operator aritmatika untuk menghitung total pinjaman dan nilai pokoknya. Total pinjaman dihitung dari jumlah pinjaman ditambah bunga (jumlah + bunga), sedangkan pokok pinjaman ditampilkan dengan mengurangi bunga dari jumlah pinjaman (jumlah - bunga). Hasilnya adalah daftar nama anggota yang memiliki pinjaman aktif, beserta nilai total utang dan nilai pokok pinjaman sebelum bunga.

Printscreen Output :



	id_anggota	nama_anggota	id_simpanan	jumlah
▶	1	Ahmad Fauzi	11	100000.00
	2	Siti Aminah	12	150000.00
	3	Rudi Hartono	13	50000.00
	4	Lilis Suryani	14	100000.00
	5	Teguh Wibowo	15	150000.00
	6	Bayu Pramana	NULL	NULL

Gambar 22. Printscreen Output Alias



	id_anggota	nama_anggota	id_simpanan	jumlah
▶	1	Ahmad Fauzi	11	100000.00
	2	Siti Aminah	12	150000.00
	3	Rudi Hartono	13	50000.00
	4	Lilis Suryani	14	100000.00
	5	Teguh Wibowo	15	150000.00

Gambar 23. Printscreen Output Operator

Penjelasan Output :

Output dari query pertama akan menampilkan data simpanan anggota koperasi dalam bentuk daftar yang mencakup nama anggota (nama_anggota), jumlah simpanan (jumlah_simpanan), dan jenis simpanan (jenis_simpanan). Data tersebut dihasilkan dari penggabungan tiga tabel yaitu simpanan, anggota, dan jenis_simpanan menggunakan klausa JOIN, sehingga hanya akan menampilkan baris data yang memiliki relasi yang cocok di ketiga tabel tersebut. Artinya, hanya anggota yang memiliki data simpanan dan jenis simpanannya yang akan ditampilkan.

Sementara itu, output dari query kedua akan menampilkan informasi pinjaman anggota yang masih berstatus "belum lunas". Kolom hasilnya berisi nama anggota (nama_anggota), total pinjaman (total_pinjaman) yang merupakan penjumlahan antara jumlah pinjaman dan bunganya, serta pokok pinjaman (pokok_pinjaman) yang merupakan hasil pengurangan antara jumlah pinjaman dan bunganya. Query ini menggabungkan tabel pinjaman dan anggota, kemudian memfilter hanya baris dengan status pinjaman "belum lunas". Outputnya akan membantu dalam memantau beban pinjaman dan sisa pokok dari anggota yang masih memiliki tanggungan.

5. Function, Grouping, Sorting**Printscreen Source Code :**

```
-- Function  
  
SELECT  
  
    UPPER(nama) AS nama_anggota_besar,  
  
    LENGTH(alamat) AS panjang_alamat  
  
FROM anggota;
```

Gambar 26. Printscreen Source Code Function

```
-- Grouping
SELECT
    id_anggota,
    SUM(jumlah) AS total_simpanan,
    COUNT(*) AS jumlah_transaksi
FROM simpanan
GROUP BY id_anggota;
```

Gambar 27. Printscreen Source Code Grouping

```
-- Sorting
SELECT
    a.nama AS nama_anggota,
    s.jumlah AS jumlah_simpanan
FROM simpanan s
JOIN anggota a ON s.id_anggota = a.id_anggota
ORDER BY s.jumlah DESC;
```

Gambar 28. Printscreen Source Code Sorting

Source Code :

```
-- Function
SELECT
    UPPER(nama) AS nama_anggota_besar,
    LENGTH(alamat) AS panjang_alamat
FROM anggota;

-- Grouping
SELECT
    id_anggota,
    SUM(jumlah) AS total_simpanan,
    COUNT(*) AS jumlah_transaksi
FROM simpanan
GROUP BY id_anggota

-- Sorting
SELECT
```



```

a.nama AS nama_anggota,
s.jumlah AS jumlah_simpanan
FROM simpanan s
JOIN anggota a ON s.id_anggota = a.id_anggota
ORDER BY s.jumlah DESC;

```

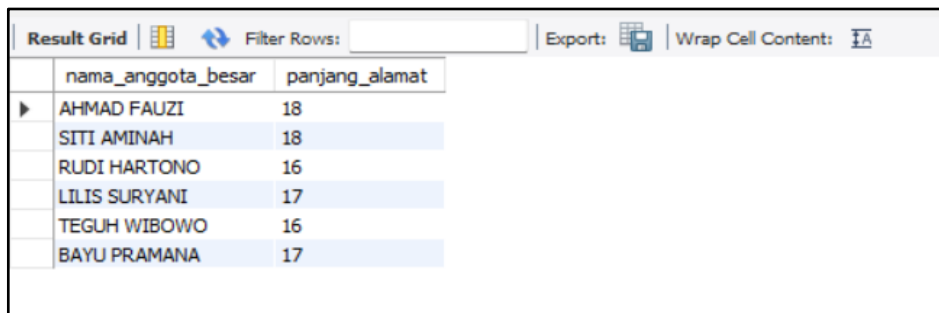
Penjelasan Source Code :

Query pertama menggunakan fungsi SQL untuk memanipulasi data dengan dua fungsi bawaan: UPPER dan LENGTH. Fungsi UPPER(nama) mengubah nilai pada kolom nama menjadi huruf kapital semua, dan hasilnya ditampilkan dengan alias nama_anggota_besar. Fungsi LENGTH(alamat) menghitung panjang karakter dalam kolom alamat dan hasilnya ditampilkan dengan alias panjang_alamat. Kedua hasil ini akan menunjukkan nama anggota dalam format huruf besar dan panjang alamat masing-masing anggota.

Query kedua menggunakan fungsi agregat dengan GROUP BY untuk mengelompokkan data berdasarkan id_anggota pada tabel simpanan. Untuk setiap grup anggota, query ini menghitung total simpanan dengan fungsi SUM(jumlah) dan jumlah transaksi dengan fungsi COUNT(*). Hasilnya akan menampilkan id_anggota beserta total simpanan yang dimiliki dan jumlah transaksi yang dilakukan oleh masing-masing anggota.

Query ketiga melakukan pengurutan data dengan menggunakan klausa ORDER BY untuk mengurutkan data berdasarkan jumlah simpanan (s.jumlah). Query ini menggabungkan tabel simpanan dan anggota untuk menampilkan nama anggota dan jumlah simpanannya. Data diurutkan dalam urutan menurun (DESC), sehingga anggota dengan jumlah simpanan terbesar akan muncul pertama dalam hasil.

Prinstscreen Output :



	nama_anggota_besar	panjang_alamat
▶	AHMAD FAUZI	18
	SITI AMINAH	18
	RUDI HARTONO	16
	LILIS SURYANI	17
	TEGUH WIBOWO	16
	BAYU PRAMANA	17

Gambar 29. Printscreen Output Membuat Function

	id_anggota	total_simpanan	jumlah_transaksi
▶	1	100000.00	1
	2	150000.00	1
	3	50000.00	1
	4	100000.00	1
	5	150000.00	1

Gambar 30. Printscreen Output Membuat Grouping

	nama_anggota	jumlah_simpanan
▶	Siti Aminah	150000.00
	Teguh Wibowo	150000.00
	Ahmad Fauzi	100000.00
	Lilis Suryani	100000.00
	Rudi Hartono	50000.00

Gambar 31. Printscreen Output Membuat Sorting

Penjelasan Output :

Query pertama menampilkan daftar anggota dengan dua kolom tambahan: nama anggota dalam huruf kapital (**UPPER(nama)**) dan panjang karakter dari alamat masing-masing (**LENGTH(alamat)**). Ini berguna untuk keperluan visualisasi dan analisis data berbasis teks. Query kedua menghitung total simpanan (**SUM(jumlah)**) dan jumlah entri simpanan (**COUNT(*)**) untuk setiap anggota berdasarkan **id_anggota**. Outputnya mengelompokkan data simpanan berdasarkan anggota dan memberikan gambaran seberapa sering serta berapa banyak simpanan yang dilakukan oleh tiap anggota. Query ketiga menampilkan nama anggota dan jumlah simpanan mereka, lalu mengurutkannya berdasarkan nilai simpanan dari yang tertinggi ke terendah (**ORDER BY s.jumlah DESC**). Output ini membantu mengidentifikasi anggota dengan nilai simpanan terbesar hingga terkecil secara langsung.

6. View, stored procedure

Printscreen Source Code :

```

33 • -- VIEW TRANSAKSI LENGKAP : Menampilkan seluruh transaksi beserta nama petugas dan anggota. --
34 create view v_transaksi_lengkap as
35 select
36     t.id_transaksi,
37     a.nama as nama_anggota,
38     p.nama as nama_petugas,
39     j.nama_transaksi,
40     t.tanggal,
41     t.jumlah,
42     t.keterangan
33 from transaksi t
34 join anggota a on t.id_anggota = a.id_anggota
35 join petugas p on t.id_petugas = p.id_petugas
36 join jenis_transaksi j on t.id_jenis_transaksi = j.id_jenis_transaksi;
37 • select * FROM v_transaksi_lengkap;

```

Gambar 32. Printscreen Source Code View

```

300 -- VIEW ANGSURAN LENGKAP : Menampilkan detail angsuran beserta data petugas dan pinjaman.
301 • create view v_angsuran_lengkap as
302 select
303     ag.id_angsuran,
304     a.nama as nama_anggota,
305     p.nama as nama_petugas,
306     ag.tanggal_angsuran,
307     ag.jumlah,
308     pin.jumlah as total_pinjaman
309 from angsuran ag
310 join pinjaman pin on ag.id_pinjaman = pin.id_pinjaman
311 join anggota a on pin.id_anggota = a.id_anggota
312 join petugas p on ag.id_petugas = p.id_petugas;
313 • select * FROM v_angsuran_lengkap;

```

Gambar 32. Printscreen Source Code View Angsuran Lengkap

```

316 -- VIEW NOTIFIKASI ANGGOTA : Menampilkan notifikasi berdasarkan anggota.--
317 • create view v_notifikasi_anggota as
318 select
319     n.id_notifikasi,
320     a.nama,
321     n.judul,
322     n.pesan,
323     n.tanggal
324 from notifikasi n
325 join anggota a on n.id_anggota = a.id_anggota;
326 • select * FROM v_notifikasi_anggota;

```

Gambar 33. Printscreen Source Code View Notifikasi Anggota

```

331 delimiter //
332 • create procedure sp_tambah_simpanan(
333     in p_id_anggota int,
334     in p_id_jenis_simpanan int,
335     in p_id_petugas int,
336     in p_tanggal date,
337     in p_jumlah decimal(15,2)
338 )
339 • begin
340     insert into simpanan(id_anggota, id_jenis_simpanan, id_petugas, tanggal, jumlah)
341     values(p_id_anggota, p_id_jenis_simpanan, p_id_petugas, p_tanggal, p_jumlah);
342 end //
343 delimiter ;
344 • call sp_tambah_simpanan(1, 2, 3, '2025-05-07', 250000.00);

```

Gambar 34. Printscreen Source Code Stored Procedure

```

348 --SP TAMBAH TRANSAKSI : Untuk mencatat transaksi baru.--
349 delimiter //
350 • create procedure sp_tambah_transaksi(
351     in p_id_anggota int,
352     in p_id_petugas int,
353     in p_id_jenis_transaksi int,
354     in p_tanggal date,
355     in p_jumlah decimal(15,2),
356     in p_keterangan text
357 )
358 • begin
359     insert into transaksi(id_anggota, id_petugas, id_jenis_transaksi, tanggal, jumlah, keterangan)
360     values(p_id_anggota, p_id_petugas, p_id_jenis_transaksi, p_tanggal, p_jumlah, p_keterangan);
361 end //
362 delimiter ;
363 • call sp_tambah_transaksi(1, 2, 4, '2025-05-07', 100000.00, 'Setoran tambahan');

```

Gambar 35. Printscreen Source Code Stored Procedure Tambah Transaksi

```

366 --SP LAPORAN SIMPAN PER ANGGOTA : Menampilkan total simpanan per anggota.--
367 delimiter //
368 • create procedure sp_laporan_simpanan_per_anggota()
369 begin
370     select
371         a.id_anggota,
372         a.nama,
373         ifnull(sum(s.jumlah), 0) as total_simpanan
374     from anggota a
375     left join simpanan s on a.id_anggota = s.id_anggota
376     group by a.id_anggota;
377 end //
378 delimiter ;
379 • call sp_laporan_simpanan_per_anggota();
380

```

Gambar 36. Printscreen Source Code SP laporan Simpan Per Anggota

```

382 --SP LAPORAN PINJAMAN LUNAS : Menampilkan data pinjaman yang sudah lunas.--
383 delimiter //
384 • create procedure sp_laporan_pinjaman_lunas()
385 begin
386     select
387         p.id_pinjaman,
388         a.nama,
389         p.jumlah,
390         p.bunga,
391         p.lama_angsuran,
392         p.status
393     from pinjaman p
394     join anggota a on p.id_anggota = a.id_anggota
395     where p.status = 'lunas';
396 end //
397 delimiter ;
398 • call sp_laporan_pinjaman_lunas();

```

Gambar 37. Printscreen Source Code Sp Laporan Pinjaman Lunas

```

401 --SP KIRIM NOTIFIKASI : Menambah notifikasi baru.--
402 delimiter //
403 • create procedure sp_kirim_notifikasi(
404     in p_id_anggota int,
405     in p_judul varchar(100),
406     in p_pesan text,
407     in p_tanggal date
408 )
409 begin
410     insert into notifikasi(id_anggota, judul, pesan, tanggal)
411     values(p_id_anggota, p_judul, p_pesan, p_tanggal);
412 end //
413 delimiter ;
414 • call sp_kirim_notifikasi(1, 'Simpanan Baru', 'Anda telah menambah simpanan sebesar Rp250.000', '2025-05-07');
415

```

Gambar 38. Printscreen Source Code SP Kirim Notifikasi

```

418 --SP DETAIL ANGSURAN BY PINJAMAN : Menampilkan angsuran berdasarkan ID pinjaman.--
419 delimiter //
420 • create procedure sp_detail_angsuran_by_pinjaman(in p_id_pinjaman int)
421 begin
422     select
423         ag.id_angsuran,
424         ag.tanggal_angsuran,
425         ag.jumlah,
426         p.nama as nama_petugas
427     from angsuran ag
428     join petugas p on ag.id_petugas = p.id_petugas
429     where ag.id_pinjaman = p_id_pinjaman;
430 end //
431 delimiter ;
432 • call sp_detail_angsuran_by_pinjaman(1);
433

```

Gambar 39. Printscreen Source Code SP Detail Angsuran

```

436 delimiter //
437 • create procedure sp_data_login_user(in p_username varchar(50))
438 begin
439     select * from login where username = p_username;
440 end //
441 delimiter ;
442 • call sp_data_login_user('ahmad');

```

Gambar 40. Printscreen Source Code Create Procedure

Source Code :

-- VIEW TRANSAKSI LENGKAP : Menampilkan seluruh transaksi beserta nama petugas dan anggota. --

create view v_transaksi_lengkap as

select

t.id_transaksi,

a.nama as nama_anggota,

p.nama as nama_petugas,

j.nama_transaksi,

t.tanggal,

t.jumlah,

t.keterangan

```

from transaksi t
join anggota a on t.id_anggota = a.id_anggota
join petugas p on t.id_petugas = p.id_petugas
join jenis_transaksi j on t.id_jenis_transaksi = j.id_jenis_transaksi;
select * FROM v_transaksi_lengkap;
-- VIEW ANGSURAN LENGKAP : Menampilkan detail angsuran beserta data
petugas dan pinjaman.
create view v_angsuran_lengkap as
select
    ag.id_angsuran,
    a.nama as nama_anggota,
    p.nama as nama_petugas,
    ag.tanggal_angsuran,
    ag.jumlah,
    pin.jumlah as total_pinjaman
from angsuran ag
join pinjaman pin on ag.id_pinjaman = pin.id_pinjaman
join anggota a on pin.id_anggota = a.id_anggota
join petugas p on ag.id_petugas = p.id_petugas;
select * FROM v_angsuran_lengkap;
-- VIEW NOTIFIKASI ANGGOTA : Menampilkan notifikasi berdasarkan
anggota.--
create view v_notifikasi_anggota as
select
    n.id_notifikasi,
    a.nama,
    n.judul,
    n.pesan,
    n.tanggal
from notifikasi n
join anggota a on n.id_anggota = a.id_anggota;
select * FROM v_notifikasi_anggota;

```

```

--SP TAMBAH SIMPANAN : Untuk menambah simpanan baru.--
delimiter //
create procedure sp_tambah_simpanan(
    in p_id_anggota int,
    in p_id_jenis_simpanan int,
    in p_id_petugas int,
    in p_tanggal date,
    in p_jumlah decimal(15,2)
)
begin
    insert into simpanan(id_anggota, id_jenis_simpanan, id_petugas, tanggal,
jumlah)
    values(p_id_anggota, p_id_jenis_simpanan, p_id_petugas, p_tanggal,
p_jumlah);
end //
delimiter ;
call sp_tambah_simpanan(1, 2, 3, '2025-05-07', 250000.00);
SELECT*FROM simpanan
--SP TAMBAH TRANSAKSI : Untuk mencatat transaksi baru.--
delimiter //
create procedure sp_tambah_transaksi(
    in p_id_anggota int,
    in p_id_petugas int,
    in p_id_jenis_transaksi int,
    in p_tanggal date,
    in p_jumlah decimal(15,2),
    in p_keterangan text
)
begin
    insert into transaksi(id_anggota, id_petugas, id_jenis_transaksi, tanggal,
jumlah, keterangan)

```



```

    values(p_id_anggota, p_id_petugas, p_id_jenis_transaksi, p_tanggal, p_jumlah,
p_keterangan);
end //
delimiter ;
call sp_tambah_transaksi(1, 2, 4, '2025-05-07', 100000.00, 'Setoran tambahan');
select*from transaksi
--SP LAPORAN SIMPAN PER ANGGOTA : Menampilkan total simpanan per
anggota.--
delimiter //
create procedure sp_laporan_simpanan_per_anggota()
begin
    select
        a.id_anggota,
        a.nama,
        ifnull(sum(s.jumlah), 0) as total_simpanan
    from anggota a
    left join simpanan s on a.id_anggota = s.id_anggota
    group by a.id_anggota;
end //
delimiter ;
call sp_laporan_simpanan_per_anggota();
--SP LAPORAN PINJAMAN LUNAS : Menampilkan data pinjaman yang sudah
lunas.--
delimiter //
create procedure sp_laporan_pinjaman_lunas()
begin
    select
        p.id_pinjaman,
        a.nama,
        p.jumlah,
        p.bunga,
        p.lama_angsuran,

```

```

        p.status
    from pinjaman p
    join anggota a on p.id_anggota = a.id_anggota
    where p.status = 'lunas';
end //
delimiter ;
call sp_laporan_pinjaman_lunas();
--SP KIRIM NOTIFIKASI : Menambah notifikasi baru.--
delimiter //
create procedure sp_kirim_notifikasi(
    in p_id_anggota int,
    in p_judul varchar(100),
    in p_pesan text,
    in p_tanggal date
)
begin
    insert into notifikasi(id_anggota, judul, pesan, tanggal)
    values(p_id_anggota, p_judul, p_pesan, p_tanggal);
end //
delimiter ;
call sp_kirim_notifikasi(1, 'Simpanan Baru', 'Anda telah menambah simpanan
sebesar Rp250.000', '2025-05-07');
--SP DETAIL ANGSURAN BY PINJAMAN : Menampilkan angsuran
berdasarkan ID pinjaman.--
delimiter //
create procedure sp_detail_angsuran_by_pinjaman(in p_id_pinjaman int)
begin
    select
        ag.id_angsuran,
        ag.tanggal_angsuran,
        ag.jumlah,
        p.nama as nama_petugas

```

```

from angsuran ag
join petugas p on ag.id_petugas = p.id_petugas
where ag.id_pinjaman = p_id_pinjaman;
end //
delimiter ;
call sp_detail_angsuran_by_pinjaman(1);
--SP DATA LOGIN user : Melihat data login berdasarkan username.--
delimiter //
create procedure sp_data_login_user(in p_username varchar(50))
begin
    select * from login where username = p_username;
end //
delimiter ;
call sp_data_login_user('ahmad');

```

Penjelasan Source Code :

Source code ini mendefinisikan tiga buah view dalam database koperasi, yang masing-masing bertujuan untuk menyederhanakan akses data dari tabel-tabel relasional dengan cara menyatukan data dari beberapa tabel terkait. View pertama adalah **v_transaksi_lengkap**, yang digunakan untuk menampilkan seluruh data transaksi beserta nama anggota, nama petugas, jenis transaksi, tanggal, jumlah, dan keterangan. View ini menggabungkan data dari tabel transaksi, anggota, petugas, dan jenis_transaksi menggunakan operasi **JOIN**, sehingga memudahkan pengguna dalam melihat informasi transaksi yang lengkap dalam satu tampilan melalui query seperti **SELECT * FROM v_transaksi_lengkap**.

View kedua adalah **v_angsuran_lengkap**, yang digunakan untuk menampilkan rincian angsuran. View ini menampilkan kolom seperti nama_anggota, nama_petugas, tanggal_angsuran, jumlah, dan total_pinjaman. Data tersebut diperoleh dengan menggabungkan tabel angsuran, pinjaman, anggota, dan petugas menggunakan **JOIN**, di mana setiap angsuran terhubung ke data pinjaman dan petugas yang memprosesnya. Dengan menggunakan view ini, pengguna bisa melihat gambaran utuh dari proses angsuran yang dilakukan anggota hanya dengan perintah **SELECT * FROM v_angsuran_lengkap**.

View ketiga adalah **v_notifikasi_anggota**, yang menyajikan daftar notifikasi berdasarkan anggota. View ini dibentuk dari notifikasi dan anggota melalui **JOIN**, dan menampilkan kolom seperti nama, judul, pesan, dan tanggal, sehingga informasi notifikasi lebih mudah dibaca dan ditelusuri. View ini sangat berguna untuk menampilkan pesan sistem atau pemberitahuan yang dikirim kepada anggota koperasi, tanpa harus melakukan query yang kompleks. Keseluruhan view ini dibuat untuk meningkatkan efisiensi dan keterbacaan data, serta dapat digunakan langsung seperti tabel biasa dengan perintah **SELECT * FROM nama_view**.

Source code di atas juga terdiri dari beberapa Stored Procedure (SP) dalam MySQL yang masing-masing dirancang untuk menangani fungsi tertentu dalam sistem koperasi, mulai dari penyimpanan data transaksi hingga pengambilan laporan. Penjelasan tiap prosedur dijabarkan secara bertahap berikut ini dalam bentuk paragraf yang menyisipkan kutipan kode seperlunya.

Prosedur pertama, **sp_tambah_simpanan**, digunakan untuk menambahkan data simpanan baru ke tabel simpanan. Dengan parameter seperti **p_id_anggota**, **p_id_jenis_simpanan**, **p_id_petugas**, **p_tanggal**, dan **p_jumlah**, prosedur ini akan menjalankan perintah **INSERT INTO simpanan(...) VALUES(...)** untuk menyimpan transaksi simpanan anggota secara otomatis. Setelah dipanggil dengan **CALL sp_tambah_simpanan(...)**, pengguna dapat memverifikasi hasilnya melalui **SELECT * FROM simpanan**.

Berikutnya, prosedur **sp_tambah_transaksi** memiliki peran serupa namun untuk mencatat transaksi ke tabel transaksi. Dengan memasukkan parameter seperti **p_id_anggota**, **p_id_petugas**, **p_id_jenis_transaksi**, dan deskripsi transaksi **p_keterangan**, sistem akan menambahkan data baru ke tabel tersebut. Penggunaan **CALL sp_tambah_transaksi(...)** dan kemudian **SELECT * FROM transaksi** memungkinkan pengguna melihat hasil pencatatan.

Prosedur **sp_laporan_simpanan_per_anggota** digunakan untuk menampilkan laporan total simpanan tiap anggota. Prosedur ini menggunakan **LEFT JOIN** antara tabel anggota dan simpanan agar anggota tanpa simpanan tetap muncul, dan **IFNULL(SUM(...), 0)** untuk menangani nilai kosong. Hasil ditampilkan melalui **CALL sp_laporan_simpanan_per_anggota()**.

Sementara itu, `sp_laporan_pinjaman_lunas` menampilkan data pinjaman yang statusnya sudah "lunas". Data ini diambil dari tabel pinjaman dan digabungkan dengan nama anggota dari tabel anggota menggunakan **JOIN**. Prosedur ini dapat digunakan oleh manajemen koperasi untuk memonitor penyelesaian pinjaman dengan menjalankan **CALL sp_laporan_pinjaman_lunas()**.

Prosedur `sp_kirim_notifikasi` digunakan untuk menambahkan pesan notifikasi ke anggota, misalnya saat ada transaksi simpanan atau perubahan status. Parameter seperti `p_judul` dan `p_pesan` diinput secara manual atau dari sistem otomatis, dan disimpan ke tabel notifikasi. Perintah pemanggilan **CALL sp_kirim_notifikasi(...)** akan langsung menambahkan baris ke tabel tersebut.

Prosedur `sp_detail_angsuran_by_pinjaman` berfungsi untuk menampilkan semua angsuran berdasarkan ID pinjaman tertentu, dengan menggabungkan data petugas yang mencatat angsuran. Prosedur ini penting untuk memeriksa detail cicilan oleh anggota tertentu dan dijalankan dengan perintah seperti **CALL sp_detail_angsuran_by_pinjaman(1)**.

Terakhir, `sp_data_login_user` dipakai untuk menampilkan data login berdasarkan username, yang penting untuk otentikasi dan pelacakan akses pengguna. Dengan menerima parameter `p_username`, prosedur ini menjalankan **SELECT * FROM login WHERE username = p_username** dan akan menampilkan data login yang sesuai saat dijalankan.

Secara keseluruhan, seluruh stored procedure ini membentuk sistem modular yang mempermudah manajemen transaksi dan informasi dalam sistem koperasi berbasis database MySQL

Prinstscreen Output :



id_transaksi	nama_anggota	nama_petugas	nama_transaksi	tanggal	jumlah	keterangan
1	Ahmad Fauzi	Rina Marlina	Pembayaran Angsuran	2025-04-10	50000.00	Angsuran ke-1
2	Siti Aminah	Budi Santoso	Penarikan Simpanan	2025-04-15	30000.00	Penarikan sukarela
3	Rudi Hartono	Eka Widya	Setoran Simpanan	2025-04-20	70000.00	Setoran bulanan
4	Lilis Suryani	Dian Prasetyo	Pembayaran Pinjaman	2025-04-25	100000.00	Pembayaran pertama
5	Teguh Wibowo	Farhan Yusuf	Pembayaran Denda	2025-05-01	10000.00	Denda keterlambatan

Gambar 41. Printscreen Output Membuat View Transaksi Lengkap

Result Grid			Filter Rows:	<input type="text"/>	Export:		Wrap Cell Content:	
	id_angsuran	nama_anggota	nama_petugas	tanggal_angsuran	jumlah	total_pinjaman		
▶	1	Ahmad Fauzi	Rina Marlina	2025-04-10	50000.00	1000000.00		
	2	Siti Aminah	Budi Santoso	2025-04-15	100000.00	2000000.00		
	3	Rudi Hartono	Eka Widya	2025-04-20	150000.00	1500000.00		
	4	Lilis Suryani	Dian Prasetyo	2025-04-25	120000.00	1200000.00		
	5	Teguh Wibowo	Farhan Yusuf	2025-05-01	180000.00	1800000.00		

Gambar 42. Printscreen Output Membuat View Angsuran Lengkap

Result Grid			Filter Rows:	<input type="text"/>	Export:		Wrap Cell Content:	
	id_notifikasi	nama	judul	pesan	tanggal			
▶	1	Ahmad Fauzi	Jatuh Tempo Angsuran	Angsuran ke-2 jatuh tempo 10 Mei	2025-05-01			
	2	Siti Aminah	Saldo Simpanan Bertambah	Penambahan simpanan Rp150.000	2025-04-10			
	3	Rudi Hartono	Pinjaman Disetujui	Pinjaman Anda telah disetujui	2025-04-01			
	4	Lilis Suryani	Denda Keterlambatan	Anda terkena denda Rp10.000	2025-05-01			
	5	Teguh Wibowo	Pembayaran Lunas	Pinjaman Anda telah lunas	2025-05-02			

Gambar 43. Printscreen Output Membuat View Notifikasi Anggota

Result Grid			Filter Rows:	<input type="text"/>	Edit:				Export/Import:			Wrap Cell Content:	
	id_simpanan	id_anggota	id_jenis_simpanan	id_petugas	tanggal	jumlah							
	1	1	1	2	2025-03-01	100000.00							
	2	2	2	3	2025-03-10	150000.00							
	3	3	3	2	2025-03-15	50000.00							
	4	4	1	1	2025-04-01	100000.00							
	5	5	2	5	2025-04-05	150000.00							
	6	1	2	3	2025-05-07	250000.00							
	NULL	NULL	NULL	NULL	NULL	NULL							

Gambar 44. Output Sp Tambah Simpanan

Result Grid							
		Filter Rows:		Edit:		Export/Import:	
	id_transaksi	id_anggota	id_petugas	id_jenis_transaksi	tanggal	jumlah	keterangan
▶	1	1	2	1	2025-04-10	50000.00	Angsuran ke-1
	2	2	3	2	2025-04-15	30000.00	Penarikan sukarela
	3	3	4	4	2025-04-20	70000.00	Setoran bulanan
	4	4	1	3	2025-04-25	100000.00	Pembayaran pertama
	5	5	5	5	2025-05-01	10000.00	Denda keterlambatan
	6	1	2	4	2025-05-07	100000.00	Setoran tambahan
transaksi 6 x							

Gambar 45. Output Sp Tambah Transaksi

Result Grid			
		Filter Rows:	
		Export:	
	id_anggota	nama	total_simpanan
▶	1	Ahmad Fauzi	350000.00
	2	Siti Aminah	150000.00
	3	Rudi Hartono	50000.00
	4	Lilis Suryani	100000.00
	5	Teguh Wibowo	150000.00
	6	Bayu Pramana	0.00

Gambar 46. Output Sp Laporan Simpan Peranggota

Result Grid						
		Filter Rows:		Export:		Wrap Cell Content: IA
	id_pinjaman	nama	jumlah	bunga	lama_angsuran	status
▶	3	Rudi Hartono	1500000.00	150000.00	6	lunas

Gambar 47. Output Sp Laporan Pinjaman Lunas

Result Grid				
		Filter Rows:		Export: IA
		Wrap Cell Content: IA		
	id_angsuran	tanggal_angsuran	jumlah	nama_petugas
▶	1	2025-04-10	50000.00	Rina Marlina

Gambar 48. Output Sp Detail Angsuran

Result Grid



Filter Rows:

Export:



Wrap Cell Content:



	id_user	username	password	role	id_petugas	id_anggota
▶	3	ahmad	1234	anggota	NULL	1

Gambar 49. Output Sp Data Login User

Penjelasan Output :

Output pertama merupakan hasil dari view bernama **v_transaksi_lengkap**, yang menampilkan seluruh data transaksi lengkap beserta nama anggota, petugas, jenis transaksi, tanggal, jumlah, dan keterangan. Misalnya, baris pertama menunjukkan bahwa anggota **Ahmad Fauzi** melakukan transaksi **Pembayaran Angsuran** pada tanggal **2025-05-01** dengan nominal sebesar **Rp500.000** yang dikategorikan sebagai **Angsuran ke-1**. View ini memudahkan pengguna untuk memahami semua transaksi yang dilakukan oleh anggota dalam satu tampilan yang terstruktur.

Output kedua adalah output dari view **v_angsuran_lengkap**, yang menampilkan informasi lengkap mengenai angsuran yang telah dilakukan oleh para anggota koperasi. Data yang ditampilkan meliputi nama anggota, nama petugas yang menangani, tanggal angsuran, jumlah angsuran, dan total pinjaman terkait. Sebagai contoh, pada baris kedua terlihat bahwa **Siti Aminah** melakukan angsuran sebesar **Rp100.000** pada tanggal **2025-04-10**, dari total pinjaman **Rp1.000.000**, dan diproses oleh **Bud Santoso**. View ini berguna untuk pelaporan angsuran secara rinci.

Output ketiga menunjukkan output dari view **v_notifikasi_anggota**, yang berfungsi untuk menampilkan daftar notifikasi kepada setiap anggota koperasi. Kolom yang ditampilkan meliputi nama anggota, judul notifikasi, isi pesan, dan tanggal notifikasi dikirimkan. Sebagai contoh, pada baris pertama, **Ahmad Fauzi** menerima notifikasi dengan judul **Jatuh Tempo Angsuran** pada **2025-05-01**, dengan pesan bahwa angsuran ke-2 jatuh tempo di tanggal tersebut. View ini memudahkan pengelola koperasi dalam menyampaikan pesan atau pengingat penting secara terpusat kepada anggota.

Output keempat menampilkan hasil dari prosedur `sp_tambah_simpanan`, di mana data simpanan anggota berhasil dimasukkan ke dalam tabel simpanan. Terlihat bahwa kolom seperti `id_simpanan`, `id_anggota`, `id_jenis_simpanan`, `id_petugas`, `tanggal`, dan `jumlah` telah terisi sesuai parameter yang diberikan, termasuk entri simpanan baru senilai Rp250.000,00 pada tanggal 2025-05-07.

Output kelima menunjukkan hasil dari `sp_tambah_transaksi`, yang mencatat transaksi baru ke dalam tabel transaksi. Kolom `id_transaksi`, `id_anggota`, `id_petugas`, `id_jenis_transaksi`, `tanggal`, `jumlah`, dan keterangan terisi lengkap. Di akhir, terlihat transaksi ke-6 berisi Setoran tambahan senilai Rp100.000,00, yang berarti prosedur dijalankan dengan sukses.

Output keenam merupakan hasil dari `sp_laporan_simpanan_per_anggota`, yang melakukan grouping berdasarkan anggota untuk menampilkan total simpanan masing-masing. Dari hasil tersebut, terlihat bahwa setiap `id_anggota` memiliki nilai `total_simpanan`, misalnya Ahmad Fauzi dengan total Rp300.000,00 dan Rudi Hartono dengan Rp150.000,00.

Output ketujuh adalah output dari `sp_laporan_pinjaman_lunas`, menampilkan informasi pinjaman yang sudah lunas. Tabel ini menampilkan data seperti `id_pinjaman`, `nama`, `jumlah`, `bunga`, `lama_angsuran`, dan `status`. Tampak bahwa pinjaman milik Rudi Hartono sebesar Rp1.500.000,00 sudah lunas.

Output kedelapan berasal dari `sp_detail_angsuran_by_pinjaman`, yang menampilkan rincian angsuran berdasarkan ID pinjaman tertentu. Output ini menunjukkan `id_angsuran`, `tanggal_angsuran`, `jumlah`, dan `nama_petugas`, di mana satu entri mencatat angsuran sebesar Rp50.000,00 oleh petugas Rina Marina pada 2025-04-10.

Terakhir, merupakan hasil dari `sp_data_login_user`, yang menampilkan data login pengguna berdasarkan username. Terlihat pengguna dengan username ahmad, password 1234, dan role anggota, terdaftar dalam sistem dengan ID user 3. Output ini menunjukkan prosedur berhasil mengekstrak data login berdasarkan parameter yang diberikan. Hal ini penting sebagai validasi bahwa stored procedure tersebut berjalan sesuai fungsi untuk proses autentikasi pengguna di sistem.

7. Nested Query, dan trigger

Printscreen Source Code :

```

239  -- nested query
240  •  SELECT
241      a.id_anggota,
242      a.nama AS nama_anggota,
243      SUM(s.jumlah) AS total_simpanan
244  FROM anggota a
245  JOIN simpanan s ON a.id_anggota = s.id_anggota
246  GROUP BY a.id_anggota, a.nama
247  HAVING SUM(s.jumlah) > (
248      SELECT AVG(total)
249      FROM (
250          SELECT id_anggota, SUM(jumlah) AS total
251          FROM simpanan
252          GROUP BY id_anggota
253      ) AS rata_simpanan
254  );

```

Gambar 50. Printscreen Source Code Nested Query

```

257  DELIMITER $$
258  •  CREATE TRIGGER after_insert_simpanan
259      AFTER INSERT ON simpanan
260      FOR EACH ROW
261      BEGIN
262          DECLARE nama_jenis VARCHAR(50);
263
264          -- Ambil nama jenis simpanan
265          SELECT nama_jenis
266          INTO nama_jenis
267          FROM jenis_simpanan
268          WHERE id_jenis_simpanan = NEW.id_jenis_simpanan;
269
270          -- Masukkan notifikasi
271          INSERT INTO notifikasi (id_anggota, judul, pesan, tanggal)
272          VALUES (
273              NEW.id_anggota,
274              'Simpanan Baru Ditambahkan',
275              CONCAT('Anda menambahkan simpanan jenis ', nama_jenis, ' sebesar Rp', NEW.jumlah),
276              CURDATE()
277          );
278      END$$
279  DELIMITER ;

```

Gambar 51. Printscreen Source Code Trigger

Source Code :

```

-- nested query
SELECT
    a.id_anggota,
    a.nama AS nama_anggota,
    SUM(s.jumlah) AS total_simpanan
FROM anggota a
JOIN simpanan s ON a.id_anggota = s.id_anggota
GROUP BY a.id_anggota, a.nama
HAVING SUM(s.jumlah) > (
    SELECT AVG(total)
    FROM (
        SELECT id_anggota, SUM(jumlah) AS total
        FROM simpanan
        GROUP BY id_anggota
    ) AS rata_simpanan
);

-- Trigger
DELIMITER $$
CREATE TRIGGER after_insert_simpanan
AFTER INSERT ON simpanan
FOR EACH ROW
BEGIN
    DECLARE nama_jenis VARCHAR(50);
    -- Ambil nama jenis simpanan
    SELECT nama_jenis
    INTO nama_jenis
    FROM jenis_simpanan
    WHERE id_jenis_simpanan = NEW.id_jenis_simpanan;
    -- Masukkan notifikasi
    INSERT INTO notifikasi (id_anggota, judul, pesan, tanggal)
    VALUES (
        NEW.id_anggota,

```

```

'Simpanan Baru Ditambahkan',
CONCAT('Anda menambahkan simpanan jenis ', nama_jenis, ' sebesar Rp',
NEW.jumlah),
CURDATE()
);
END$$
DELIMITER ;

```

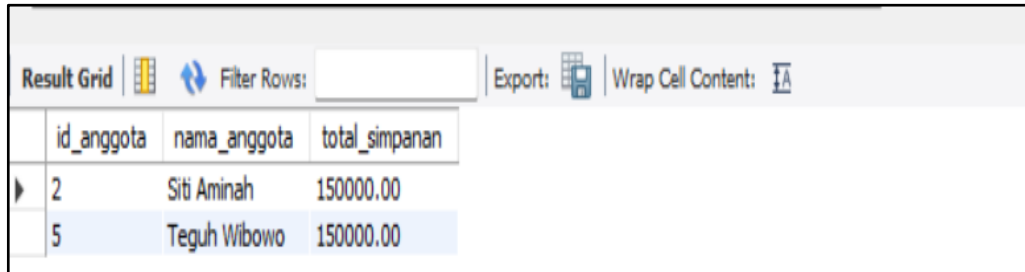
Penjelasan Source Code :

Source code di atas terdiri dari dua bagian, yaitu nested query dan trigger, yang masing-masing berperan untuk melakukan analisis data dan otomatisasi pencatatan notifikasi. Bagian pertama adalah nested query (kueri bersarang) yang digunakan untuk menampilkan daftar anggota yang memiliki total simpanan di atas rata-rata seluruh anggota. Kueri utama melakukan penggabungan (**JOIN**) antara tabel anggota dan simpanan berdasarkan `id_anggota`, lalu menghitung jumlah total simpanan masing-masing anggota dengan fungsi **SUM(s.jumlah)** dan mengelompokkan hasilnya berdasarkan `a.id_anggota` dan `a.nama`. Bagian pentingnya adalah klausa **HAVING**, yang menyaring hasil hanya untuk anggota yang total simpanannya lebih besar dari rata-rata simpanan seluruh anggota. Rata-rata ini dihitung oleh subquery: **SELECT AVG(total) FROM (...)**, di mana subquery terdalam menghitung total simpanan per anggota (**SUM(jumlah) per id_anggota**), lalu hasil tersebut dirata-ratakan menggunakan **AVG**. Konstruksi ini berguna untuk membandingkan performa simpanan tiap anggota terhadap rata-rata keseluruhan.

Bagian kedua adalah kode trigger bernama `after_insert_simpanan`, yang dirancang untuk berjalan otomatis setelah data baru dimasukkan ke tabel simpanan. Trigger ini bertugas untuk menghasilkan notifikasi kepada anggota yang bersangkutan. Di dalam blok **BEGIN ... END**, pertama-tama dideklarasikan variabel `nama_jenis`, lalu dilakukan pengambilan nama jenis simpanan dari tabel `jenis_simpanan` menggunakan pernyataan **SELECT ... INTO**. Setelah informasi tersebut diperoleh, sistem secara otomatis akan menyisipkan (**INSERT**) satu baris notifikasi ke dalam tabel notifikasi, dengan judul 'Simpanan Baru Ditambahkan' dan pesan yang dibentuk menggunakan **CONCAT** untuk menggabungkan jenis simpanan dan jumlah simpanan baru yang dimasukkan. Fungsi **CURDATE()** digunakan untuk

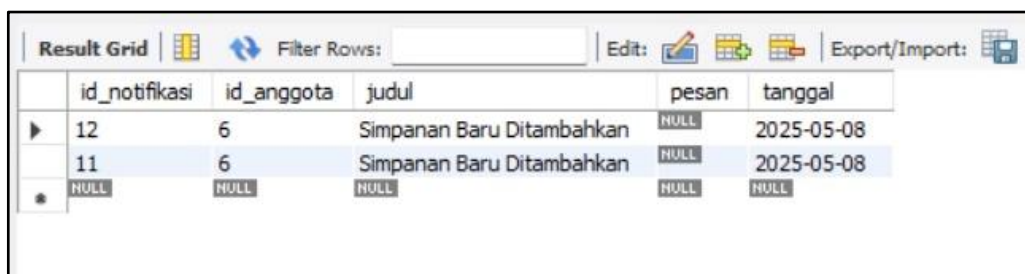
mencatat tanggal saat notifikasi dibuat. Trigger ini berguna untuk memberi tahu anggota secara otomatis tanpa perlu dilakukan secara manual oleh petugas.

Prinstscreen Output :



	id_anggota	nama_anggota	total_simpanan
▶	2	Siti Aminah	150000.00
	5	Teguh Wibowo	150000.00

Gambar 52. Printscreen Output *Nested Query*



	id_notifikasi	id_anggota	judul	pesan	tanggal
▶	12	6	Simpanan Baru Ditambahkan	NULL	2025-05-08
	11	6	Simpanan Baru Ditambahkan	NULL	2025-05-08
✱	NULL	NULL	NULL	NULL	NULL

Gambar 53. Printscreen Output Setelah Insert Trigger

Penjelasan Output :

Output dari query nested yang menampilkan daftar anggota yang memiliki total simpanan lebih besar dari rata-rata total simpanan seluruh anggota. Berdasarkan hasil tersebut, hanya dua anggota yang ditampilkan, yaitu Siti Aminah dan Teguh Wibowo, masing-masing dengan total simpanan sebesar Rp150.000.00. Hal ini menunjukkan bahwa total simpanan kedua anggota tersebut melebihi nilai rata-rata yang dihitung dari total simpanan seluruh anggota lainnya. Query ini menggunakan subquery untuk menghitung rata-rata total simpanan setiap anggota, kemudian outer query membandingkannya dengan total simpanan tiap anggota dan hanya menampilkan yang lebih besar dari rata-rata. Dengan demikian, output ini memberikan informasi penting mengenai anggota dengan tingkat simpanan yang berada di atas rata-rata keseluruhan.

Trigger ini dibuat untuk secara otomatis menambahkan notifikasi setiap kali ada data baru dimasukkan ke tabel simpanan. Trigger ini dijalankan setelah (**AFTER**) baris baru ditambahkan (**INSERT**) ke tabel simpanan. Ketika trigger aktif Ia mengambil nama_jenis dari tabel jenis_simpanan berdasarkan id_jenis_simpanan yang sama seperti yang baru dimasukkan (**NEW.id_jenis_simpanan**). Lalu ia menyusun sebuah pesan notifikasi, misalnya:

“Anda menambahkan simpanan jenis Wajib sebesar Rp75000”. Setelah itu, ia memasukkan pesan tersebut ke tabel notifikasi, beserta id_anggota, judul pesan, dan tanggal hari ini (CURDATE()).

8. DCL

Printscreen Source Code :

```
257 DELIMITER $$
258 • CREATE TRIGGER after_insert_simpanan
259 AFTER INSERT ON simpanan
260 FOR EACH ROW
261 BEGIN
262     DECLARE nama_jenis VARCHAR(50);
263
264     -- Ambil nama jenis simpanan
265     SELECT nama_jenis
266     INTO nama_jenis
267     FROM jenis_simpanan
268     WHERE id_jenis_simpanan = NEW.id_jenis_simpanan;
269
270     -- Masukkan notifikasi
271     INSERT INTO notifikasi (id_anggota, judul, pesan, tanggal)
272     VALUES (
273         NEW.id_anggota,
274         'Simpanan Baru Ditambahkan',
275         CONCAT('Anda menambahkan simpanan jenis ', nama_jenis, ' sebesar Rp', NEW.jumlah),
276         CURDATE()
277     );
278 END$$
279 DELIMITER ;
```

Gambar 54. Printscreen Source Code Trigger

Source Code :

-- Buat user anggota/petugas (hak terbatas)

```
CREATE USER 'app_user'@'localhost' IDENTIFIED BY 'user123_';
```

```
GRANT SELECT, INSERT, UPDATE ON koperasi.login TO 'app_user'@'localhost';
```

```
GRANT SELECT, INSERT, UPDATE ON koperasi.simpanan TO 'app_user'@'localhost';
```

```
GRANT SELECT, INSERT, UPDATE ON koperasi.transaksi TO 'app_user'@'localhost';
```

```
GRANT SELECT, INSERT ON koperasi.pinjaman TO 'app_user'@'localhost';
```

```
GRANT SELECT ON koperasi.notifikasi TO 'app_user'@'localhost';
```

-- Buat user admin (akses penuh)

```
CREATE USER 'admin_user'@'localhost' IDENTIFIED BY 'admin123_';  
GRANT ALL PRIVILEGES ON koperasi.* TO 'admin_user'@'localhost' WITH  
GRANT OPTION;
```

-- Terapkan perubahan hak akses

```
FLUSH PRIVILEGES;
```

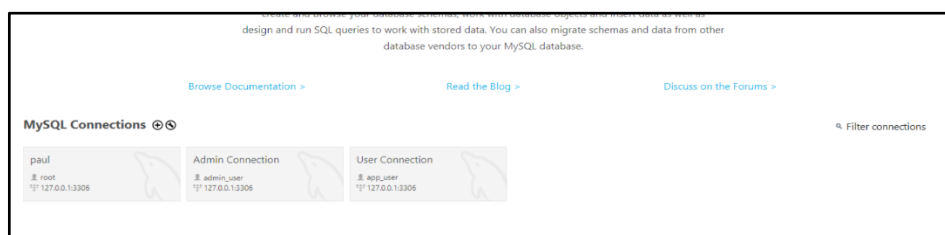
```
SELECT user, host FROM mysql.user;
```

Penjelasan Source Code :

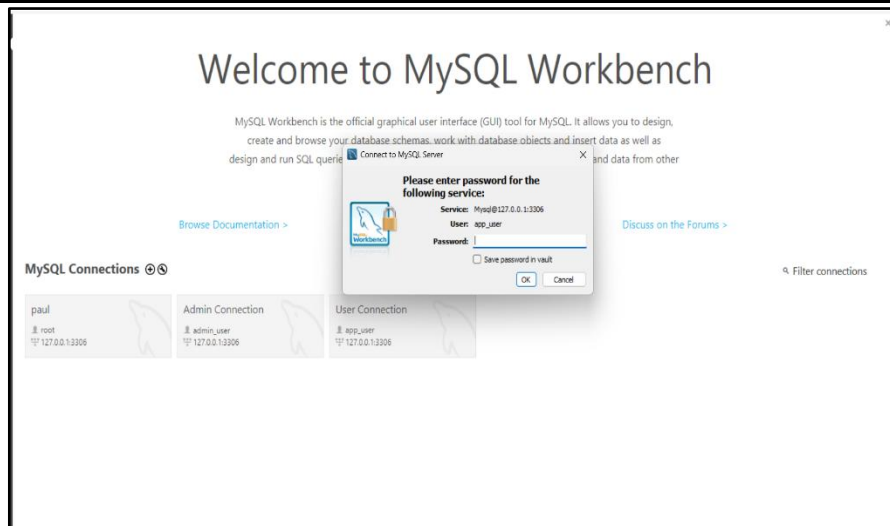
Kode SQL di atas digunakan untuk mengelola hak akses pengguna dalam sistem basis data MySQL pada skema (database) bernama koperasi. Pertama-tama, kode membuat akun pengguna terbatas bernama app_user yang hanya dapat diakses dari localhost, dengan kata sandi user123_. Pengguna ini diberi izin terbatas, yaitu hanya dapat melakukan operasi **SELECT**, **INSERT**, dan **UPDATE** pada tabel login, simpanan, dan transaksi, serta hanya **SELECT** dan **INSERT** pada tabel pinjaman, dan hanya **SELECT** pada tabel notifikasi. Tujuannya adalah membatasi hak pengguna biasa (anggota atau petugas) agar tidak bisa menghapus atau memodifikasi data sensitif secara bebas.

Selanjutnya, dibuat juga akun admin admin_user, juga hanya dari localhost, dengan kata sandi admin123_. Pengguna ini diberikan akses penuh (**ALL PRIVILEGES**) terhadap seluruh tabel dalam database koperasi, termasuk hak untuk memberikan akses (**WITH GRANT OPTION**) kepada pengguna lain. Setelah pemberian hak akses, perintah **FLUSH PRIVILEGES** dijalankan untuk memastikan semua perubahan hak akses diterapkan oleh MySQL. Terakhir, perintah **SELECT user, host FROM mysql.user;** digunakan untuk menampilkan daftar pengguna dan host mereka dari sistem database, guna memverifikasi bahwa pengguna yang baru telah dibuat dan terdaftar dengan benar.

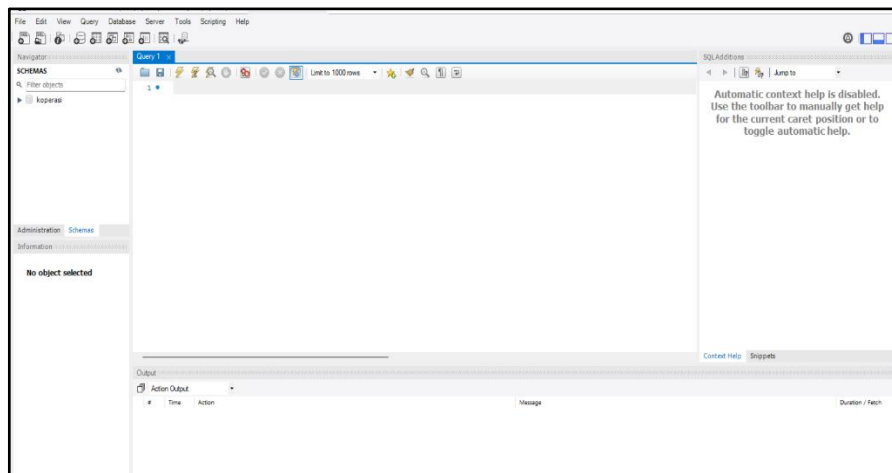
Printscreen Output :



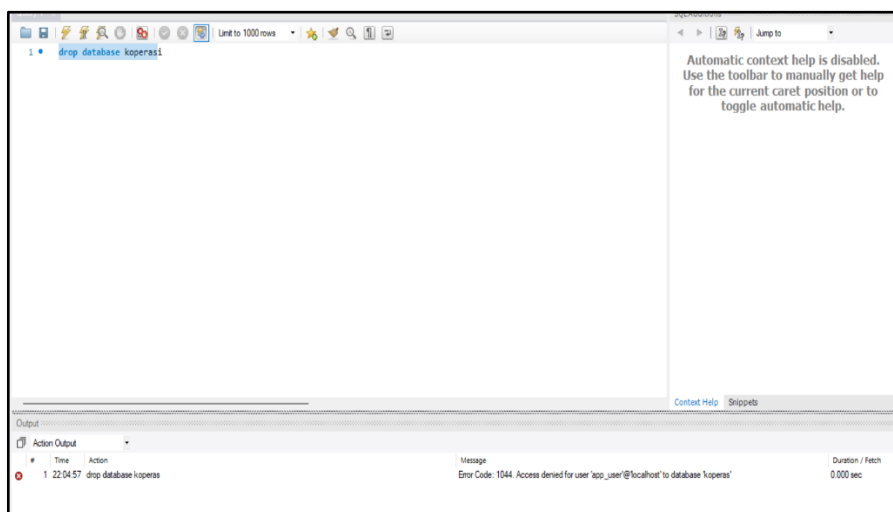
Gambar 55. Printscreen Admin dan User



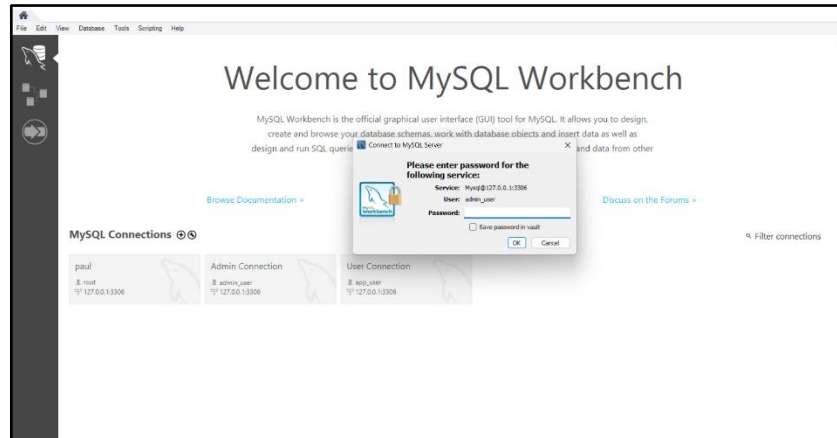
Gambar 56 Printscreen Output User



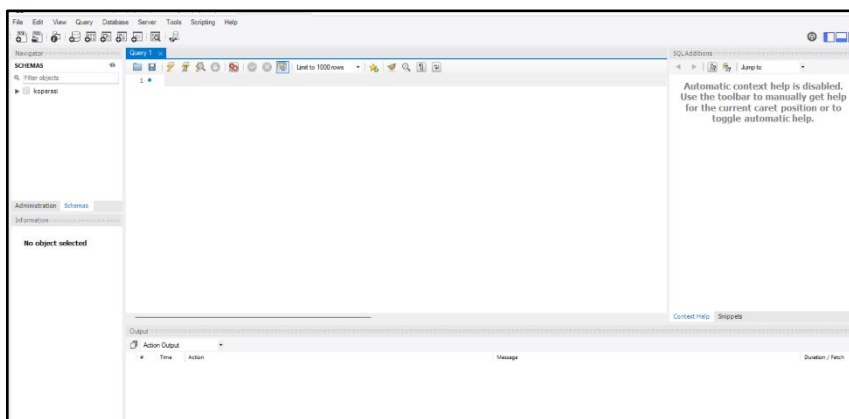
Gambar 57. Printscreen Output Halaman User



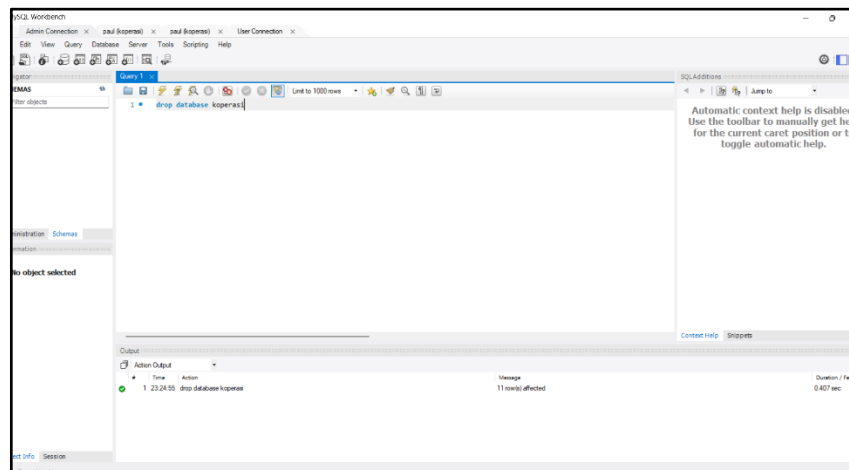
Gambar 58. Printscreen Output User Mencoba Hapus Database



Gambar 59. Printscreen Output Admin



Gambar 60. Printscreen Output Halaman Admin



Gambar 61. Printscreen Output Admin Mencoba Hapus Database

Penjelasan Output :

Setelah seluruh perintah dijalankan, output yang dihasilkan berasal dari perintah `SELECT user, host FROM mysql.user;`. Output ini menampilkan daftar semua pengguna yang terdaftar dalam sistem MySQL beserta asal host yang

digunakan untuk mengakses database. Tabel hasil memiliki dua kolom utama, yaitu user dan host, yang menunjukkan nama pengguna dan dari mana pengguna tersebut diizinkan mengakses server database. Dalam kasus ini, dua pengguna baru akan terlihat dalam hasil: `app_user` dan `admin_user`, masing-masing hanya diizinkan mengakses dari `localhost`. Pengguna lain yang sudah ada sebelumnya, seperti `root`, juga akan ikut ditampilkan dalam hasil tersebut.

Ketika pengguna masuk ke halaman admin atau user dalam aplikasi, perilaku mereka terhadap database akan berbeda sesuai hak akses yang telah diberikan. Jika dari halaman user (yang menggunakan akun `app_user`) dilakukan percobaan untuk menghapus database atau menjalankan perintah `DROP`, maka tindakan tersebut akan gagal atau ditolak oleh sistem. Hal ini terjadi karena `app_user` hanya memiliki hak terbatas seperti `SELECT`, `INSERT`, dan `UPDATE` pada beberapa tabel tertentu, tanpa akses `DELETE`, `DROP`, atau `ALTER`. Sebaliknya, jika tindakan tersebut dilakukan dari halaman admin (yang menggunakan akun `admin_user`), maka penghapusan database dapat berhasil dilakukan karena `admin_user` memiliki `ALL PRIVILEGES`, termasuk hak untuk menghapus (`DROP`) database, tabel, atau objek lainnya. Dengan demikian, pembatasan hak akses ini memastikan keamanan sistem dengan memisahkan wewenang antara pengguna biasa dan administrator.

Sebaliknya, saat login ke halaman admin menggunakan akun `admin_user`, pengguna ini diberikan akses penuh terhadap semua tabel dan operasi di dalam database koperasi. Dengan hak istimewa `ALL PRIVILEGES` yang disertai opsi `WITH GRANT OPTION`, admin dapat melakukan segala bentuk manipulasi data dan struktur, termasuk membuat atau menghapus tabel, serta bahkan menghapus keseluruhan database. Artinya, jika admin memilih menjalankan perintah seperti `DROP DATABASE koperasi`;, perintah itu akan berhasil dijalankan tanpa penolakan, karena admin secara eksplisit diberi wewenang penuh. Admin juga dapat memberikan atau mencabut hak akses terhadap user lain, memperluas atau membatasi kontrol keamanan sistem sesuai kebijakan pengelolaan. Perbedaan hak akses ini sangat penting dalam menjaga integritas dan keamanan sistem. Dengan membatasi hak user biasa, risiko kesalahan fatal akibat tindakan tidak sengaja—seperti menghapus data penting—dapat dihindari.

BAB IV

PENUTUP

a. Kesimpulan

Pengembangan sistem manajemen simpan pinjam koperasi merupakan langkah strategis untuk meningkatkan efisiensi dan efektivitas pengelolaan administrasi koperasi. Dengan sistem terkomputerisasi, proses pencatatan simpanan, pinjaman, dan angsuran dapat dilakukan secara otomatis, mengurangi kesalahan manual, serta mempercepat pelayanan kepada anggota. Sistem ini juga memberikan kemudahan dalam pembuatan laporan keuangan yang akurat dan *real-time*, sehingga mendukung pengambilan keputusan yang lebih tepat oleh pengurus koperasi. Selain itu, transparansi dan akuntabilitas pengelolaan keuangan koperasi dapat meningkat, yang pada akhirnya memperkuat kepercayaan anggota terhadap koperasi. Dengan demikian, penerapan sistem manajemen simpan pinjam berbasis teknologi informasi sangat penting untuk mendukung keberlangsungan dan pengembangan koperasi di era digital.

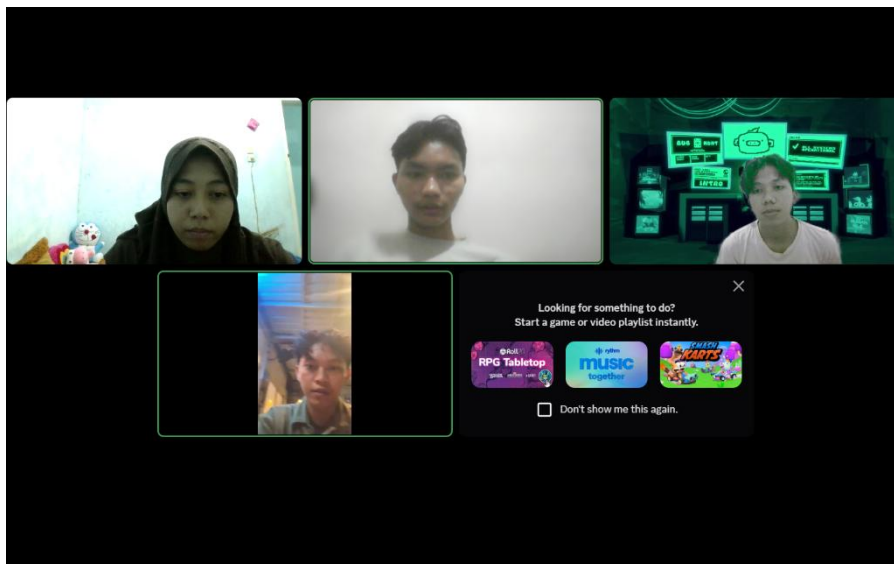
b. Saran

Dalam rangka mengoptimalkan penggunaan sistem manajemen simpan pinjam koperasi, disarankan agar implementasi dilakukan secara bertahap agar proses adaptasi oleh pengurus dan anggota dapat berjalan lancar tanpa mengganggu operasional koperasi. Selain itu, integrasi dengan sistem pembayaran digital sangat penting guna mempermudah proses transaksi simpan dan pinjam secara lebih efisien dan praktis. Penting juga untuk menyediakan fasilitas *backup* data secara berkala agar data-data penting koperasi aman dari risiko kehilangan akibat kerusakan sistem atau *human error*. Pelatihan yang memadai perlu diberikan kepada seluruh pengguna sistem agar mereka mampu mengoperasikan sistem dengan efektif dan meminimalisir kesalahan dalam pengelolaan data. Selanjutnya, penting bagi koperasi untuk melakukan pemeliharaan dan pengembangan sistem secara berkala agar sistem tetap sesuai dengan kebutuhan dan perkembangan teknologi terkini. Terakhir, koperasi harus memberikan perhatian khusus pada aspek keamanan data dengan menerapkan protokol keamanan yang kuat.

DAFTAR PUSTAKA

- Dwi Praba, A., & Safitri, M. (n.d.). *STUDI PERBANDINGAN PERFORMANSI ANTARA MYSQL DAN POSTGRESQL. VIII(2)*. <https://www.adminer.org/>.
- Hasan¹, L., Riphoh, :, & Perkasa, D. (2023). PERAN KOPERASI SIMPAN PINJAM DALAM MEMBERDAYAKAN EKONOMI MASYARAKAT (STUDI PADA KSP SURYA ABADI MANDIRI, MEDAN KRIO, KECAMATAN SUNGGAL, KABUPATEN DELI SERDANG. *GENTA MULIA-JURNAL ILMIAH PENDIDIKAN*, 3(1), 319–327.
- Juswadi, J., & Sumarna, P. (2023). Perkembangan Usaha Koperasi Simpan Pinjam Indonesia dan Faktor yang Mempengaruhinya Periode 2013-2020. *Paspalum: Jurnal Ilmiah Pertanian*, 11(1), 74. <https://doi.org/10.35138/paspalum.v11i1.541>
- Kalsum Siregar, U., Arbaim Sitakar, T., Haramain, S., Nur Salamah Lubis, Z., Nadhirah, U., & Sains dan Teknologi, F. (2024). *Pengembangan database Management system menggunakan My SQL* (Vol. 1, Issue 1).
- Setiyadi, D., & Nidaul Khasanah, F. (2019). Data Manipulation Language (DML) Database Penjadwalan Dosen menggunakan SQL Server. *Bina Insani ICT Journal*, 6(2), 145–154.
- Sinata, F. (2023). ANALISA PERBANDINGAN TINGKAT EFISIENSI ALGORITMA DATA DEFINITION LANGUAGE (DDL) COPY, INPLACE, INSTANT DATABASE MYSQL Comparison Analysis Of Efficiency Level Of Mysql Data Definition Language Algorithm Copy, Inplace, Instant Database. *Jurnal Algoritma, Logika Dan Komputasi*, VI(1), 503–508. <https://doi.org/10.30813/j-alu.v2i2.4322>
- Wijaya, T. A., Menteng, C., Surya, A., Julianto, A., & Utami, E. (2021). *PERANCANGAN DESAIN BASIS DATA SISTEM INFORMASI GEOGRAFIS TANAH PENDUDUK DENGAN MENERAPKAN MODEL DATA RELASIONAL (STUDI KASUS: DESA TUMBANG MANTUHE KABUPATEN GUNUNG MAS PROVINSI KALIMANTAN TENGAH)*. 15(1).
- Wongso, F. (2015). Speizer et al 2001.pdf. *Jurnal Ilmiah EkoWongso, F. (2015). Speizer et al 2001.Pdf. Jurnal Ilmiah Ekonomi Dan Bisnis*, 12(1), 46–60. *Nomi Dan Bisnis*, 12(1), 46–60.

Dokumentasi Pengerjaan





**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

PRAKTIKUM PROYEK BASIS DATA LANJUT

Nama Mahasiswa : 1. Rayhan Muhammad Adha (G1A023051)
2. Ayu Dewanti Suci (G1A023057)
3. Mohammad Irvan Ramadhansyah (G1A023089)
4. Hendro Paulus Limbong (G1A023091)

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

Asisten : 1. Reksi Hendra Pratama (G1A021032)
2. Merly Yuni Purnama (G1A022006)
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)
8. Dian Ardiyanti Saputri (G1A022084)

Laporan Tugas Besar	Catatan dan Tanda Tangan
Laporan Tugas Besar	

