

**LAPORAN TUGAS BESAR
PROYEK BASIS DATA LANJUT**



Disusun Oleh :

Kelompok 4

Anggota Kelompok

1. Ghazi Al-Ghifari (G1A023053)
2. Rahman Firdaus Illahi (G1A023055)
3. Merischa Theresia Hutaaruk (G1A023071)
4. Salah Nasser Hasan Meqdam (G1A023095)

Nama Asisten Dosen :

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

Dosen Pengampu :

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

**PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS BENGKULU**

2025

KATA PENGANTAR

Syukur alhamdulillah kami panjatkan kepada Allah Swt. atas rahmat dan hidayah-Nya yang memungkinkan kami menyelesaikan tugas besar ini tepat pada waktunya. Laporan ini disusun sebagai pemenuhan tugas besar dalam mata kuliah Proyek Basis Data Lanjut.

Tugas ini bukan hanya menguji kemampuan akademis kami, namun juga mengembangkan keterampilan dalam bekerja sama, riset, dan presentasi. Kami berharap laporan ini dapat memberikan pemahaman yang komprehensif dan mendalam mengenai isu yang kami bahas, sekaligus menawarkan solusi yang kami rancang untuk menghadapi tantangan yang ada. Laporan ini disusun dengan sistematis agar pembaca dapat mengikuti alur pemikiran kami dengan mudah.

Kami juga ingin mengucapkan terima kasih yang sebesar-besarnya kepada asisten dosen dan rekan-rekan yang telah memberikan bimbingan, dukungan, dan masukan yang sangat berharga. Keterlibatan mereka memperkaya diskusi dan memberikan perspektif tambahan yang sangat berarti. Dengan penuh kebanggaan, kami menyerahkan laporan ini sebagai hasil dari perjalanan kami dalam mata kuliah ini yang penuh dedikasi dan semangat.

Semoga laporan ini dapat memberikan manfaat dan inspirasi bagi setiap pembaca. Terima kasih atas kesempatan yang diberikan, dan kami berharap laporan ini dapat berkontribusi secara bermakna untuk pemahaman lebih lanjut di mata kuliah ini.

Bengkulu, 5 Mei 2025

Penulis

DAFTAR ISI

KATA PENGANTAR	1
DAFTAR ISI.....	2
DAFTAR GAMBAR	3
DAFTAR TABEL.....	4
LANDASAN TEORI.....	5
HASIL DAN PEMBAHASAN.....	8
KESIMPULAN DAN SARAN.....	44
DAFTAR PUSTAKA	45
JOBDESK	47
DOKUMENTASI Pengerjaan.....	48

DAFTAR GAMBAR

Gambar 1. ERD	8
Gambar 2. Source Code Database dan Tabel	9
Gambar 3. Source Code Tabel (1).....	10
Gambar 4. Source Code Tabel (2).....	11
Gambar 5. Source Code Tabel (3).....	12
Gambar 6. Input Data (1)	14
Gambar 7. Input Data (2)	15
Gambar 8. Input Data (3)	15
Gambar 9. Input Data (4)	15
Gambar 22. Alias dan Operator	23
Gambar 24. Join	24
Gambar 30. Function.....	28
Gambar 31. Penggunaan Function	29
Gambar 33. Grouping dan Sorting	30
Gambar 35. View	31
Gambar 36. Memanggil View	32
Gambar 37. Stored Procedure	33
Gambar 38. Penggunaan Stored Procedure.....	34
Gambar 40. Nested Query.....	36
Gambar 42. Trigger.....	37
Gambar 43. Penggunaan Trigger	38
Gambar 46. DCL.....	39
Gambar 47. Admin dan User	41
Gambar 48. User	41
Gambar 49. Halaman User.....	42
Gambar 50. User Mencoba Menghapus Database	42
Gambar 51. Admin.....	42
Gambar 52. Halaman Admin (1).....	42
Gambar 53. Admin Berhasil Menghapus Database	43

DAFTAR TABEL

Gambar 10. Tabel Pengguna	20
Gambar 11. Tabel Pengeluaran	20
Gambar 12. Tabel Pemasukan	20
Gambar 13. Tabel Hutang	21
Gambar 14. Tabel Tabungan.....	21
Gambar 15. Tabel Asset.....	21
Gambar 16. Tabel Laporan	21
Gambar 17. Tabel Mencatat Pengeluaran	22
Gambar 18. Tabel Mencatat Pemasukan.....	22
Gambar 19. Tabel Mencatat Hutang	22
Gambar 20. Tabel Mencatat Asset	22
Gambar 21. Tabel Mencatat Tabungan.....	23
Gambar 23. Tabel Output Alias dan Operator	24
Gambar 25. Tabel Output Inner Join.....	26
Gambar 26. Tabel Output Left Join	27
Gambar 27. Tabel Output Right Join	27
Gambar 28. Tabel Output Full Join.....	27
Gambar 29. Tabel Output Cross Join.....	27
Gambar 32. Tabel Output Function	30
Gambar 34. Tabel Output Grouping dan Sorting.....	31
Gambar 39. Tabel Output Stored Procedure	35
Gambar 41. Tabel Output Nested Query.....	36
Gambar 44. Output Sebelum Dilakukan Insert	39
Gambar 45. Output Setelah Dilakukan Insert	39

LANDASAN TEORI

Basis data merupakan kumpulan informasi yang diorganisasikan secara sistematis dan terstruktur, memungkinkan penyimpanan, pengambilan, serta pengelolaan data secara efisien. Sistem ini dirancang untuk menangani berbagai jenis informasi yang saling berhubungan, sekaligus menyediakan mekanisme pengaksesan yang teratur bagi pengguna maupun aplikasi perangkat lunak.

Untuk mengelola basis data secara efektif, digunakan Sistem Manajemen Basis Data (DBMS) yang berfungsi sebagai lapisan antarmuka antara pengguna dengan data yang tersimpan. DBMS menyediakan berbagai fitur penting seperti kontrol akses, mekanisme backup dan recovery, serta kemampuan untuk memanipulasi data dalam jumlah besar dengan performa optimal. Sistem ini menjadi tulang punggung dalam pengelolaan informasi di era digital saat ini.

Salah satu jenis DBMS yang paling banyak digunakan adalah RDBMS (Relational Database Management System), yang mengorganisasikan data dalam bentuk tabel-tabel yang saling berelasi. Beberapa contoh implementasi RDBMS yang populer antara lain Oracle Database, Microsoft SQL Server, MySQL, IBM DB2, dan Microsoft Access.

Bahasa pemrograman SQL (Structured Query Language) telah menjadi standar de facto dalam berinteraksi dengan sistem basis data relasional. SQL menyediakan sintaks yang relatif mudah dipahami namun sangat powerful untuk melakukan berbagai operasi seperti query, update, dan manipulasi struktur data. Kemampuannya yang komprehensif membuat SQL didukung oleh hampir semua platform database modern.

MySQL, sebagai salah satu implementasi RDBMS open source terkemuka, memiliki sejarah perkembangan yang menarik. Bermula dari proyek UNIREG yang dikembangkan oleh Michael "Monty" Widenius di Swedia, MySQL kini telah tumbuh menjadi solusi database yang handal.

Dengan model lisensi ganda (GPL untuk penggunaan gratis dan lisensi komersial untuk kebutuhan khusus), MySQL menawarkan fleksibilitas penggunaan baik untuk proyek pribadi maupun enterprise. Fitur-fitur unggulannya seperti performa tinggi, skalabilitas, dan kompatibilitas dengan berbagai platform membuatnya menjadi pilihan populer di kalangan pengembang.

Klausa SQL JOIN merupakan operasi fundamental dalam SQL yang digunakan untuk menggabungkan data dari dua atau lebih tabel berdasarkan kolom terkait. Operasi ini bekerja dengan mencocokkan nilai-nilai kunci antara tabel, memungkinkan pengguna untuk menganalisis data relasional secara terpadu. JOIN sangat penting dalam sistem basis data relasional karena memungkinkan pembuatan laporan yang membutuhkan data dari beberapa tabel sekaligus.

Terdapat beberapa jenis JOIN yang masing-masing memiliki karakteristik unik. INNER JOIN hanya menampilkan baris yang memiliki kecocokan di kedua tabel, sehingga cocok untuk analisis data yang memerlukan relasi eksak. LEFT JOIN (atau LEFT OUTER JOIN) mengambil semua data dari tabel kiri dan hanya data yang cocok dari tabel kanan, berguna ketika ingin mempertahankan semua record dari tabel utama meskipun tidak ada relasi. Sebaliknya, RIGHT JOIN mengambil semua data dari tabel kanan dan hanya data yang cocok dari tabel kiri. Sementara itu, FULL JOIN menggabungkan semua data dari kedua tabel dan mengisi dengan NULL jika tidak ada kecocokan, menjadikannya solusi komprehensif untuk analisis data terpadu.

Mekanisme alias SQL memainkan peran penting dalam meningkatkan keterbacaan dan pengelolaan kueri Anda. Inti dari fitur ini adalah AS kata kunci, yang merupakan alat yang ampuh untuk menetapkan nama sementara pada tabel, kolom, dan bahkan subkueri.

Alias SQL adalah nama sementara yang ditetapkan pada tabel atau kolom dalam kueri. Alias umumnya digunakan untuk membuat nama kolom dan tabel lebih mudah dibaca atau ringkas, atau untuk mengatasi konflik penamaan saat menggabungkan beberapa tabel.

Sorting merupakan fungsi dasar dalam pemrosesan data yang mengatur elemen secara berurutan, baik ascending maupun descending. Dalam sistem basis data, sorting membantu pengelompokan data berdasarkan kriteria tertentu seperti kode pos atau kategori usia.

Berbagai algoritma sorting seperti Quick Sort, Merge Sort, Insertion Sort, dan Selection Sort telah dikembangkan untuk kebutuhan berbeda. Sementara itu, pengelompokan data dalam SQL menggunakan klausa GROUP BY yang mengatur baris dengan nilai identik dalam kolom tertentu, menciptakan kelompok data unik.

Fungsi agregat seperti SUM, COUNT, atau AVG bekerja dengan GROUP BY untuk menghasilkan nilai tunggal dari sekumpulan baris, dan dapat digunakan dalam SELECT, ORDER BY, maupun HAVING. Klausa HAVING khususnya berfungsi untuk menyaring hasil agregasi, berbeda dengan WHERE yang menyaring baris individual. Tanpa GROUP BY, fungsi agregat akan diterapkan pada seluruh baris dalam tabel, memberikan fleksibilitas dalam analisis data.

Fungsi CONCAT dalam SQL digunakan untuk menggabungkan beberapa string menjadi satu nilai, dengan mengabaikan argumen NULL sehingga hasil penggabungan tetap konsisten. Fungsi ini sangat berguna untuk menyajikan data dalam format yang lebih informatif dan mudah dibaca, seperti menggabungkan nama depan dan belakang atau membuat pesan yang mengandung nilai variabel.

Selain CONCAT, SQL juga menyediakan fitur VIEW yang berfungsi sebagai tabel virtual berbasis hasil query. VIEW tidak menyimpan data fisik, melainkan hanya menyimpan definisi query SELECT, sehingga menghemat ruang penyimpanan sekaligus mempertahankan struktur data asli.

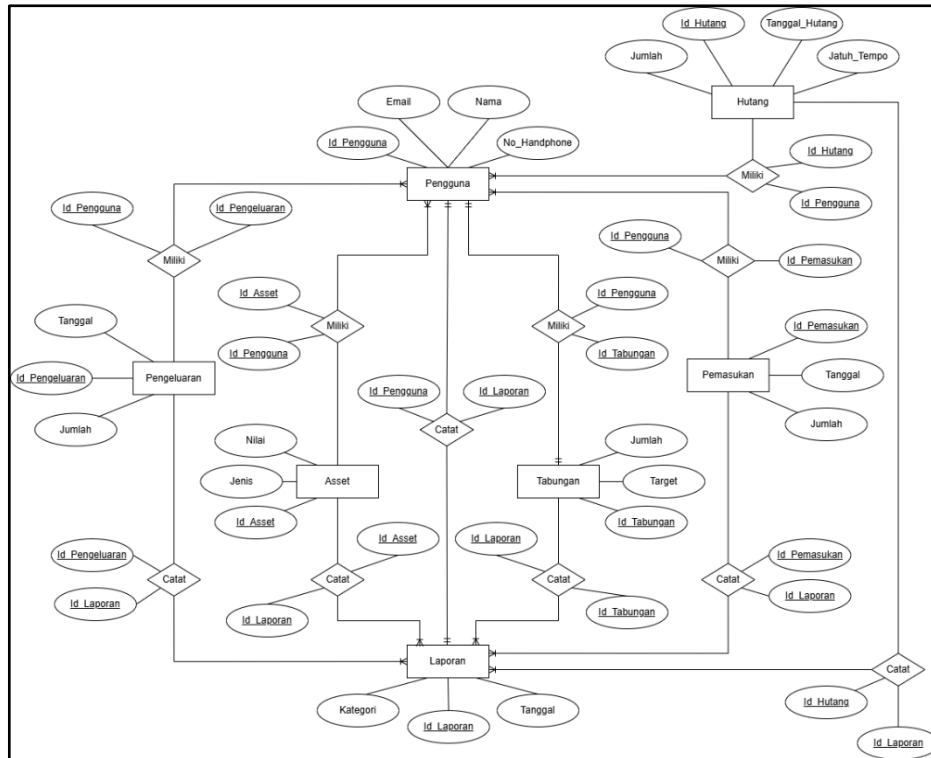
Penggunaan VIEW memberikan beberapa keuntungan penting seperti peningkatan keamanan data dengan membatasi akses ke tabel dasar, peningkatan efisiensi query yang kompleks, serta optimasi penggunaan memori karena tidak perlu menyimpan data duplikat. Fitur-fitur ini menjadikan SQL tetap relevan sebagai bahasa query utama untuk pengolahan data bisnis secara real-time hingga saat ini.

Trigger merupakan sekumpulan perintah atau sintaks yang akan secara otomatis dijalankan jika terjadi operasi tertentu dalam tabel atau view. Trigger merupakan sebuah blok PL/SQL atau prosedur PL/SQL yang berhubungan dengan table, view, schema atau database.

Trigger Akan dieksekusi secara implicit pada saat sebuah kejadian tertentu terjadi. Digunakan untuk memanggil satu atau beberapa perintah SQL secara otomatis sebelum atau sesudah terjadi proses INSERT, UPDATE atau DELETE dari suatu tabel.

Dalam praktiknya, trigger sering dimanfaatkan untuk menerapkan business rule, menjaga integritas data, membuat audit trail, atau memicu proses otomatis lainnya yang terkait dengan perubahan data.

HASIL DAN PEMBAHASAN



Gambar 1. ERD

Penjelasan :

Entity Relationship Diagram (ERD) yang ditampilkan menggambarkan struktur basis data untuk Sistem Manajemen Keuangan Pribadi. Dalam sistem ini, entitas utama adalah Pengguna, yang menjadi pusat hubungan terhadap berbagai entitas lain, seperti Pengeluaran, Pemasukan, Tabungan, Hutang, Aset, dan Laporan.

Entitas Pengguna memiliki atribut utama seperti Id_Pengguna, Nama, Email, dan No_Handphone. Pengguna dapat memiliki (relasi "Miliki") beberapa entitas lain. Misalnya, pengguna dapat memiliki banyak Pengeluaran, yang dicatat melalui entitas Pengeluaran dengan atribut Id_Pengeluaran, Tanggal, dan Jumlah. Pengeluaran ini selanjutnya bisa dicatat dalam Laporan, yang memiliki atribut seperti Id_Laporan, Tanggal, dan Kategori.

Begitu pula dengan Pemasukan, pengguna dapat memiliki beberapa pemasukan dengan atribut Id_Pemasukan, Tanggal, dan Jumlah. Pemasukan ini juga dapat dicatat dalam Laporan. Hubungan antara entitas-entitas ini dengan Laporan divisualisasikan melalui relasi "Catat", yang menunjukkan bahwa berbagai transaksi keuangan (pengeluaran, pemasukan, tabungan, hutang, dan aset) semuanya dapat menjadi bagian dari laporan yang dibuat oleh pengguna.

Entitas Tabungan memiliki atribut Id_Tabungan, Jumlah, dan Target. Pengguna juga bisa memiliki lebih dari satu tabungan, dan setiap tabungan dapat dicatat ke dalam laporan. Demikian pula, entitas Hutang yang dimiliki oleh pengguna mencakup atribut Id_Hutang, Jumlah, Tanggal_Hutang, dan Jatuh_Tempo.

Selain itu, pengguna dapat memiliki berbagai Aset, yang dijelaskan dengan atribut Id_Aset, Jenis, dan Nilai. Sama seperti entitas lain, aset juga memiliki hubungan "Catat" dengan laporan, memungkinkan pengguna untuk mencatat informasi aset dalam laporan keuangan mereka.

Struktur ERD ini menggunakan relasi many-to-many yang difasilitasi oleh hubungan "Miliki" dan "Catat", serta menyediakan fleksibilitas tinggi dalam pengelolaan data keuangan pribadi. Semua entitas saling terhubung melalui entitas Laporan, yang berfungsi sebagai ringkasan atau catatan keseluruhan aktivitas keuangan pengguna. Ini menunjukkan bahwa sistem dirancang untuk memberikan pelacakan dan dokumentasi menyeluruh terhadap setiap aspek keuangan pengguna.

1. Membuat Database dan Tabel

Printscreen Source Code :

```
1 • CREATE DATABASE financial_management;
2 • USE financial_management;
3
4 • CREATE TABLE Pengguna (
5     Id_Pengguna INT PRIMARY KEY,
6     Nama VARCHAR(100),
7     Email VARCHAR(100),
8     No_Handphone VARCHAR(20)
9 );
10
11 • CREATE TABLE Pengeluaran (
12     Id_Pengeluaran INT PRIMARY KEY,
13     Id_Pengguna INT,
14     Tanggal DATE,
15     Jumlah DECIMAL(12,2),
16     FOREIGN KEY (Id_Pengguna) REFERENCES Pengguna(Id_Pengguna)
17 );
18
19 • CREATE TABLE Pemasukan (
20     Id_Pemasukan INT PRIMARY KEY,
21     Id_Pengguna INT,
22     Tanggal DATE,
23     Jumlah DECIMAL(12,2),
24     FOREIGN KEY (Id_Pengguna) REFERENCES Pengguna(Id_Pengguna)
25 );
```

Gambar 2. Source Code Database dan Tabel

Source Code :

CREATE DATABASE financial_management;

USE financial_management;

CREATE TABLE Pengguna (

Id_Pengguna INT PRIMARY KEY,

```

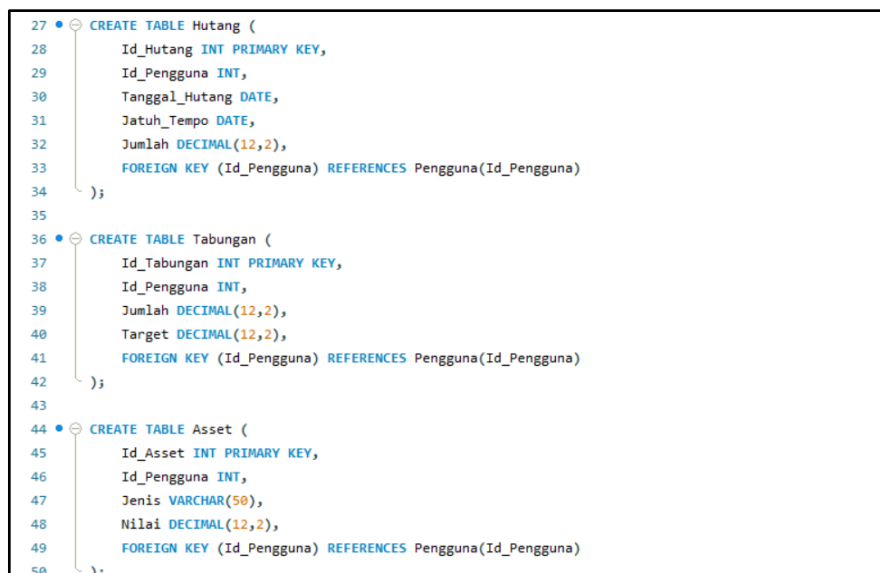
Nama VARCHAR(100),
Email VARCHAR(100),
No_Handphone VARCHAR(20)
);

CREATE TABLE Pengeluaran (
    Id_Pengeluaran INT PRIMARY KEY,
    Id_Pengguna INT,
    Tanggal DATE,
    Jumlah DECIMAL(12,2),
    FOREIGN KEY (Id_Pengguna) REFERENCES Pengguna(Id_Pengguna)
);

CREATE TABLE Pemasukan (
    Id_Pemasukan INT PRIMARY KEY,
    Id_Pengguna INT,
    Tanggal DATE,
    Jumlah DECIMAL(12,2),
    FOREIGN KEY (Id_Pengguna) REFERENCES Pengguna(Id_Pengguna)
);

```

Printscreen Source Code :



```

27 • CREATE TABLE Hutang (
28     Id_Hutang INT PRIMARY KEY,
29     Id_Pengguna INT,
30     Tanggal_Hutang DATE,
31     Jatuh_Tempo DATE,
32     Jumlah DECIMAL(12,2),
33     FOREIGN KEY (Id_Pengguna) REFERENCES Pengguna(Id_Pengguna)
34 );
35
36 • CREATE TABLE Tabungan (
37     Id_Tabungan INT PRIMARY KEY,
38     Id_Pengguna INT,
39     Jumlah DECIMAL(12,2),
40     Target DECIMAL(12,2),
41     FOREIGN KEY (Id_Pengguna) REFERENCES Pengguna(Id_Pengguna)
42 );
43
44 • CREATE TABLE Asset (
45     Id_Asset INT PRIMARY KEY,
46     Id_Pengguna INT,
47     Jenis VARCHAR(50),
48     Nilai DECIMAL(12,2),
49     FOREIGN KEY (Id_Pengguna) REFERENCES Pengguna(Id_Pengguna)
50 );

```

Gambar 3. Source Code Tabel (1)

Source Code :

```

CREATE TABLE Hutang (

```

```

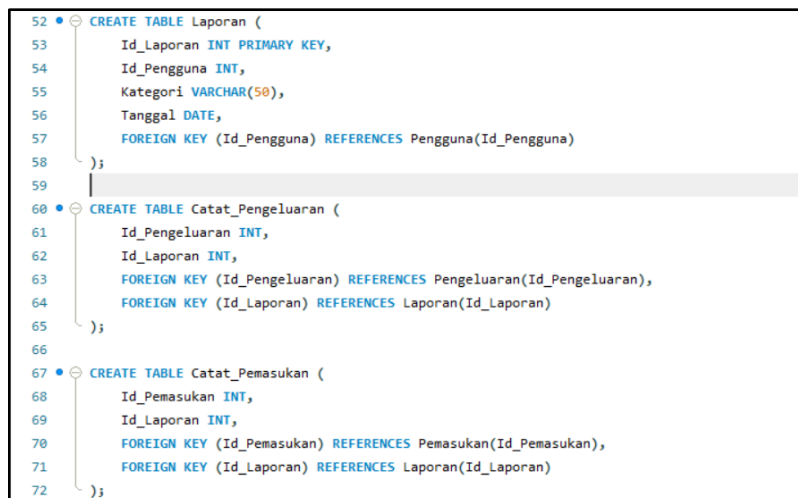
Id_Hutang INT PRIMARY KEY,
Id_Pengguna INT,
Tanggal_Hutang DATE,
Jatuh_Tempo DATE,
Jumlah DECIMAL(12,2),
FOREIGN KEY (Id_Pengguna) REFERENCES Pengguna(Id_Pengguna)
);

CREATE TABLE Tabungan (
    Id_Tabungan INT PRIMARY KEY,
    Id_Pengguna INT,
    Jumlah DECIMAL(12,2),
    Target DECIMAL(12,2),
    FOREIGN KEY (Id_Pengguna) REFERENCES Pengguna(Id_Pengguna)
);

CREATE TABLE Asset (
    Id_Asset INT PRIMARY KEY,
    Id_Pengguna INT,
    Jenis VARCHAR(50),
    Nilai DECIMAL(12,2),
    FOREIGN KEY (Id_Pengguna) REFERENCES Pengguna(Id_Pengguna)
);

```

Printscreen Source Code :



```

52 • CREATE TABLE Laporan (
53     Id_Laporan INT PRIMARY KEY,
54     Id_Pengguna INT,
55     Kategori VARCHAR(50),
56     Tanggal DATE,
57     FOREIGN KEY (Id_Pengguna) REFERENCES Pengguna(Id_Pengguna)
58 );
59
60 • CREATE TABLE Catat_Pengeluaran (
61     Id_Pengeluaran INT,
62     Id_Laporan INT,
63     FOREIGN KEY (Id_Pengeluaran) REFERENCES Pengeluaran(Id_Pengeluaran),
64     FOREIGN KEY (Id_Laporan) REFERENCES Laporan(Id_Laporan)
65 );
66
67 • CREATE TABLE Catat_Pemasukan (
68     Id_Pemasukan INT,
69     Id_Laporan INT,
70     FOREIGN KEY (Id_Pemasukan) REFERENCES Pemasukan(Id_Pemasukan),
71     FOREIGN KEY (Id_Laporan) REFERENCES Laporan(Id_Laporan)
72 );

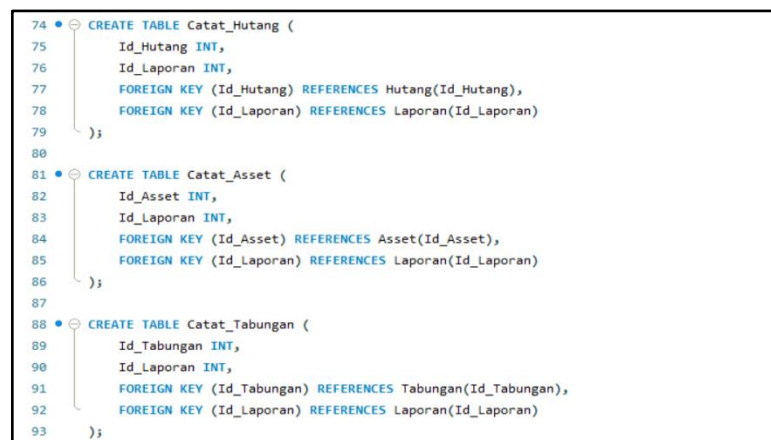
```

Gambar 4. Source Code Tabel (2)

Source Code :

```
CREATE TABLE Laporan (  
    Id_Laporan INT PRIMARY KEY,  
    Id_Pengguna INT,  
    Kategori VARCHAR(50),  
    Tanggal DATE,  
    FOREIGN KEY (Id_Pengguna) REFERENCES Pengguna(Id_Pengguna)  
);  
  
CREATE TABLE Catat_Pengeluaran (  
    Id_Pengeluaran INT,  
    Id_Laporan INT,  
    FOREIGN KEY (Id_Pengeluaran) REFERENCES  
Pengeluaran(Id_Pengeluaran),  
    FOREIGN KEY (Id_Laporan) REFERENCES Laporan(Id_Laporan)  
);  
  
CREATE TABLE Catat_Pemasukan (  
    Id_Pemasukan INT,  
    Id_Laporan INT,  
    FOREIGN KEY (Id_Pemasukan) REFERENCES  
Pemasukan(Id_Pemasukan),  
    FOREIGN KEY (Id_Laporan) REFERENCES Laporan(Id_Laporan)  
);
```

Printscreen Source Code :



```
74 CREATE TABLE Catat_Hutang (  
75     Id_Hutang INT,  
76     Id_Laporan INT,  
77     FOREIGN KEY (Id_Hutang) REFERENCES Hutang(Id_Hutang),  
78     FOREIGN KEY (Id_Laporan) REFERENCES Laporan(Id_Laporan)  
79 );  
80  
81 CREATE TABLE Catat_Asset (  
82     Id_Asset INT,  
83     Id_Laporan INT,  
84     FOREIGN KEY (Id_Asset) REFERENCES Asset(Id_Asset),  
85     FOREIGN KEY (Id_Laporan) REFERENCES Laporan(Id_Laporan)  
86 );  
87  
88 CREATE TABLE Catat_Tabungan (  
89     Id_Tabungan INT,  
90     Id_Laporan INT,  
91     FOREIGN KEY (Id_Tabungan) REFERENCES Tabungan(Id_Tabungan),  
92     FOREIGN KEY (Id_Laporan) REFERENCES Laporan(Id_Laporan)  
93 );
```

Gambar 5. Source Code Tabel (3)

Source Code :

```
CREATE TABLE Catat_Hutang (  
    Id_Hutang INT,  
    Id_Laporan INT,  
    FOREIGN KEY (Id_Hutang) REFERENCES Hutang(Id_Hutang),  
    FOREIGN KEY (Id_Laporan) REFERENCES Laporan(Id_Laporan)  
);  
  
CREATE TABLE Catat_Asset (  
    Id_Asset INT,  
    Id_Laporan INT,  
    FOREIGN KEY (Id_Asset) REFERENCES Asset(Id_Asset),  
    FOREIGN KEY (Id_Laporan) REFERENCES Laporan(Id_Laporan)  
);  
  
CREATE TABLE Catat_Tabungan (  
    Id_Tabungan INT,  
    Id_Laporan INT,  
    FOREIGN KEY (Id_Tabungan) REFERENCES Tabungan(Id_Tabungan),  
    FOREIGN KEY (Id_Laporan) REFERENCES Laporan(Id_Laporan)  
);
```

Penjelasan Source Code :

Kode SQL di atas mendefinisikan struktur database bernama `financial_management`, yang bertujuan untuk mengelola data keuangan pengguna secara terorganisir. Pertama, perintah `CREATE DATABASE` membuat database baru, kemudian `USE financial_management` mengaktifkan database tersebut agar semua tabel yang dibuat berikutnya berada di dalamnya.

Selanjutnya, beberapa tabel utama didefinisikan untuk merepresentasikan berbagai entitas keuangan. Tabel Pengguna menyimpan data dasar pengguna, seperti ID (sebagai PRIMARY KEY), nama, email, dan nomor handphone. Setiap pengguna dapat memiliki beberapa entri pada tabel-tabel lainnya yang berhubungan dengan aktivitas keuangannya.

Tabel Pengeluaran dan Pemasukan merekam masing-masing pengeluaran dan pemasukan pengguna. Keduanya memiliki kolom `Id_Pengguna` sebagai foreign key

yang merujuk ke Pengguna, sehingga tiap transaksi terhubung dengan pemiliknya. Kolom Tanggal dan Jumlah mencatat kapan transaksi terjadi dan nominalnya.

Tabel Hutang menyimpan informasi pinjaman atau utang yang dimiliki pengguna, termasuk tanggal transaksi (Tanggal_Hutang), tanggal jatuh tempo (Jatuh_Tempo), dan jumlah yang dipinjam. Sama seperti sebelumnya, Id_Pengguna digunakan untuk mengaitkan data utang ke pengguna terkait.

Tabel Tabungan mencatat berapa banyak uang yang disimpan (Jumlah) serta target tabungan (Target) untuk masing-masing pengguna. Sementara itu, tabel Asset menyimpan data aset pengguna, termasuk jenis aset (misalnya properti, kendaraan, dsb.) dan nilai keuangannya.

Tabel Laporan berfungsi sebagai entitas pusat untuk menyusun laporan keuangan pengguna, berisi kolom Kategori untuk jenis laporan, serta tanggal pembuatannya. Setiap laporan juga terkait dengan pengguna.

Bagian terakhir dari kode ini menciptakan tabel-tabel penghubung (Catat_Pengeluaran, Catat_Pemasukan, Catat_Hutang, Catat_Asset, dan Catat_Tabungan) yang berfungsi mencatat hubungan antara entri transaksi spesifik dan laporan keuangan tertentu. Setiap tabel ini mengandung dua kolom, yaitu ID dari transaksi spesifik dan ID dari laporan yang mencatat transaksi tersebut.

2. Meng-input Data

Printscreen Source Code :

```
1 • INSERT INTO Pengguna VALUES
2 (1, 'Ghazi', 'ghazi@mail.com', '08123456789'),
3 (2, 'Rayhan', 'rayhan@mail.com', '08123456780'),
4 (3, 'Agyl', 'agyl@mail.com', '08123456781'),
5 (4, 'Aulia', 'aulia@mail.com', '08123456782'),
6 (5, 'Dimas', 'dimas@mail.com', '08123456783'),
7 (6, 'Edo', 'edo@mail.com', '08123456784'),
8 (7, 'Salah', 'salah@mail.com', '08123456785'),
9 (8, 'Juan', 'juan@mail.com', '08123456786'),
10 (9, 'Memei', 'memei@mail.com', '08123456787'),
11 (10, 'Rahman', 'rahman@mail.com', '08123456788');
12
13 INSERT INTO Pengeluaran VALUES
14 (1, 1, '2025-05-01', 500000),
15 (2, 2, '2025-05-02', 400000),
16 (3, 3, '2025-05-03', 700000),
17 (4, 4, '2025-05-04', 600000),
18 (5, 5, '2025-05-05', 300000),
19 (6, 6, '2025-05-06', 200000),
20 (7, 7, '2025-05-07', 100000),
21 (8, 8, '2025-05-08', 900000),
22 (9, 9, '2025-05-09', 850000),
23 (10, 10, '2025-05-10', 750000);
```

Gambar 6. Input Data (1)

```

25 • INSERT INTO Pemasukan VALUES
26 (1, 1, '2025-05-02', 3000000),
27 (2, 2, '2025-05-02', 2500000),
28 (3, 3, '2025-05-02', 2000000),
29 (4, 4, '2025-05-02', 1800000),
30 (5, 5, '2025-05-02', 2200000),
31 (6, 6, '2025-05-02', 2700000),
32 (7, 7, '2025-05-02', 1500000),
33 (8, 8, '2025-05-02', 1900000),
34 (9, 9, '2025-05-02', 3100000),
35 (10, 10, '2025-05-02', 3300000);
36
37 • INSERT INTO Hutang VALUES
38 (1, 1, '2025-05-03', '2025-06-01', 1000000),
39 (2, 2, '2025-05-03', '2025-06-01', 500000),
40 (3, 3, '2025-05-03', '2025-06-01', 750000),
41 (4, 4, '2025-05-03', '2025-06-01', 650000),
42 (5, 5, '2025-05-03', '2025-06-01', 900000),
43 (6, 6, '2025-05-03', '2025-06-01', 300000),
44 (7, 7, '2025-05-03', '2025-06-01', 450000),
45 (8, 8, '2025-05-03', '2025-06-01', 800000),
46 (9, 9, '2025-05-03', '2025-06-01', 700000),
47 (10, 10, '2025-05-03', '2025-06-01', 600000);

```

Gambar 7. Input Data (2)

```

49 • INSERT INTO Tabungan VALUES
50 (1, 1, 1500000, 5000000),
51 (2, 2, 1400000, 4000000),
52 (3, 3, 1300000, 3000000),
53 (4, 4, 1600000, 3500000),
54 (5, 5, 1700000, 3700000),
55 (6, 6, 1800000, 3600000),
56 (7, 7, 1200000, 3200000),
57 (8, 8, 1100000, 3300000),
58 (9, 9, 1900000, 3900000),
59 (10, 10, 2000000, 4000000);
60
61 • INSERT INTO Asset VALUES
62 (1, 1, 'Motor', 12000000),
63 (2, 2, 'Laptop', 8000000),
64 (3, 3, 'Handphone', 4000000),
65 (4, 4, 'Sepeda', 3000000),
66 (5, 5, 'TV', 3500000),
67 (6, 6, 'PC', 7000000),
68 (7, 7, 'Tablet', 4500000),
69 (8, 8, 'Camera', 6000000),
70 (9, 9, 'Jam Tangan', 2000000),
71 (10, 10, 'Kulkas', 5000000);

```

Gambar 8. Input Data (3)

```

73 • INSERT INTO Laporan VALUES
74 (1, 1, 'Bulanan', '2025-05-04'),
75 (2, 2, 'Bulanan', '2025-05-04'),
76 (3, 3, 'Bulanan', '2025-05-04'),
77 (4, 4, 'Bulanan', '2025-05-04'),
78 (5, 5, 'Bulanan', '2025-05-04'),
79 (6, 6, 'Bulanan', '2025-05-04'),
80 (7, 7, 'Bulanan', '2025-05-04'),
81 (8, 8, 'Bulanan', '2025-05-04'),
82 (9, 9, 'Bulanan', '2025-05-04'),
83 (10, 10, 'Bulanan', '2025-05-04');
84
85 • INSERT INTO Catat_Pengeluaran VALUES
86 (1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), (10, 10);
87 • INSERT INTO Catat_Pemasukan VALUES
88 (1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), (10, 10);
89 • INSERT INTO Catat_Hutang VALUES
90 (1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), (10, 10);
91 • INSERT INTO Catat_Tabungan VALUES
92 (1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), (10, 10);
93 • INSERT INTO Catat_Asset VALUES
94 (1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), (10, 10);

```

Gambar 9. Input Data (4)

Source Code :**INSERT INTO Pengguna VALUES**

```
(1, 'Ghazi', 'ghazi@mail.com', '08123456789'),  
(2, 'Rayhan', 'rayhan@mail.com', '08123456780'),  
(3, 'Agyl', 'agyl@mail.com', '08123456781'),  
(4, 'Aulia', 'aulia@mail.com', '08123456782'),  
(5, 'Dimas', 'dimas@mail.com', '08123456783'),  
(6, 'Edo', 'edo@mail.com', '08123456784'),  
(7, 'Salah', 'salah@mail.com', '08123456785'),  
(8, 'Juan', 'juan@mail.com', '08123456786'),  
(9, 'Memei', 'memei@mail.com', '08123456787'),  
(10, 'Rahman', 'rahman@mail.com', '08123456788');
```

INSERT INTO Pengeluaran VALUES

```
(1, 1, '2025-05-01', 500000),  
(2, 2, '2025-05-02', 400000),  
(3, 3, '2025-05-03', 700000),  
(4, 4, '2025-05-04', 600000),  
(5, 5, '2025-05-05', 300000),  
(6, 6, '2025-05-06', 200000),  
(7, 7, '2025-05-07', 100000),  
(8, 8, '2025-05-08', 900000),  
(9, 9, '2025-05-09', 850000),  
(10, 10, '2025-05-10', 750000);
```

INSERT INTO Pemasukan VALUES

```
(1, 1, '2025-05-02', 3000000),  
(2, 2, '2025-05-02', 2500000),  
(3, 3, '2025-05-02', 2000000),  
(4, 4, '2025-05-02', 1800000),  
(5, 5, '2025-05-02', 2200000),  
(6, 6, '2025-05-02', 2700000),
```

```
(7, 7, '2025-05-02', 1500000),  
(8, 8, '2025-05-02', 1900000),  
(9, 9, '2025-05-02', 3100000),  
(10, 10, '2025-05-02', 3300000);
```

INSERT INTO Hutang VALUES

```
(1, 1, '2025-05-03', '2025-06-01', 1000000),  
(2, 2, '2025-05-03', '2025-06-01', 500000),  
(3, 3, '2025-05-03', '2025-06-01', 750000),  
(4, 4, '2025-05-03', '2025-06-01', 650000),  
(5, 5, '2025-05-03', '2025-06-01', 900000),  
(6, 6, '2025-05-03', '2025-06-01', 300000),  
(7, 7, '2025-05-03', '2025-06-01', 450000),  
(8, 8, '2025-05-03', '2025-06-01', 800000),  
(9, 9, '2025-05-03', '2025-06-01', 700000),  
(10, 10, '2025-05-03', '2025-06-01', 600000);
```

INSERT INTO Tabungan VALUES

```
(1, 1, 1500000, 5000000),  
(2, 2, 1400000, 4000000),  
(3, 3, 1300000, 3000000),  
(4, 4, 1600000, 3500000),  
(5, 5, 1700000, 3700000),  
(6, 6, 1800000, 3600000),  
(7, 7, 1200000, 3200000),  
(8, 8, 1100000, 3300000),  
(9, 9, 1900000, 3900000),  
(10, 10, 2000000, 4000000);
```

INSERT INTO Asset VALUES

```
(1, 1, 'Motor', 12000000),  
(2, 2, 'Laptop', 8000000),
```

```
(3, 3, 'Handphone', 4000000),  
(4, 4, 'Sepeda', 3000000),  
(5, 5, 'TV', 3500000),  
(6, 6, 'PC', 7000000),  
(7, 7, 'Tablet', 4500000),  
(8, 8, 'Camera', 6000000),  
(9, 9, 'Jam Tangan', 2000000),  
(10, 10, 'Kulkas', 5000000);
```

INSERT INTO Laporan VALUES

```
(1, 1, 'Bulanan', '2025-05-04'),  
(2, 2, 'Bulanan', '2025-05-04'),  
(3, 3, 'Bulanan', '2025-05-04'),  
(4, 4, 'Bulanan', '2025-05-04'),  
(5, 5, 'Bulanan', '2025-05-04'),  
(6, 6, 'Bulanan', '2025-05-04'),  
(7, 7, 'Bulanan', '2025-05-04'),  
(8, 8, 'Bulanan', '2025-05-04'),  
(9, 9, 'Bulanan', '2025-05-04'),  
(10, 10, 'Bulanan', '2025-05-04');
```

INSERT INTO Catat_Pengeluaran VALUES

```
(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), (10, 10);
```

INSERT INTO Catat_Pemasukan VALUES

```
(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), (10, 10);
```

INSERT INTO Catat_Hutang VALUES

```
(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), (10, 10);
```

INSERT INTO Catat_Tabungan VALUES

```
(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), (10, 10);
```

INSERT INTO Catat_Asset VALUES

```
(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), (10, 10);
```

Penjelasan Source Code :

Kode dimulai dengan memasukkan data 10 pengguna ke dalam tabel Pengguna. Setiap pengguna memiliki ID unik (1-10), nama, alamat email dengan format nama@mail.com, dan nomor handphone yang berurutan dari '08123456789' hingga '08123456788'. Data pengguna ini mencakup nama-nama seperti Ghazi, Rayhan, Agyl, hingga Rahman. Setiap record dibangun dengan pola konsisten dimana email menggunakan format nama pengguna dan domain mail.com, sementara nomor telepon mengikuti pola berurutan yang mudah diidentifikasi. Data pengguna ini akan menjadi basis untuk semua transaksi dan catatan keuangan berikutnya.

Untuk tabel Pengeluaran, terdapat 10 entri yang sesuai dengan masing-masing pengguna. Pengeluaran dicatat dengan tanggal mulai 1 Mei 2025 hingga 10 Mei 2025, dengan nilai yang bervariasi dari Rp100.000 hingga Rp900.000. Nilai pengeluaran tertinggi dimiliki oleh Juan (Rp900.000) dan terendah oleh Salah (Rp100.000). Di sisi Pemasukan, semua transaksi masuk dicatat pada tanggal 2 Mei 2025 dengan nilai lebih besar daripada pengeluaran, berkisar antara Rp1.500.000 hingga Rp3.300.000. Pemasukan tertinggi dimiliki oleh Rahman dan terendah oleh Salah. Pola ini menunjukkan bahwa semua pengguna memiliki siklus pendapatan di tanggal yang sama tetapi dengan nilai yang berbeda-beda sesuai kapasitas finansial masing-masing.

Data Hutang menunjukkan bahwa semua pengguna melakukan pinjaman pada 3 Mei 2025 dengan jatuh tempo 1 Juni 2025. Nilai hutang bervariasi dari Rp300.000 hingga Rp1.000.000. Untuk Tabungan, setiap pengguna memiliki jumlah tabungan saat ini dan target tabungan. Jumlah tabungan saat ini berkisar antara Rp1.100.000 hingga Rp2.000.000, sementara target tabungan antara Rp3.000.000 hingga Rp5.000.000. Data ini menggambarkan profil menabung yang berbeda-beda di antara pengguna, dengan beberapa seperti Rahman sudah mencapai setengah dari targetnya.

Tabel Asset mencatat berbagai jenis kepemilikan pengguna dengan nilai berbeda. Untuk sistem pelaporan, dibuat 10 laporan bulanan di tanggal 4 Mei 2025, masing-masing untuk setiap pengguna. Setiap laporan kemudian dihubungkan dengan semua transaksi dan catatan keuangan pengguna tersebut melalui tabel-tabel penghubung. Ini menciptakan hubungan one-to-one antara setiap transaksi dengan

laporan bulanan masing-masing pengguna.

Output :

	Id_Pengguna	Nama	Email	No_Handphone
▶	1	Ghazi	ghazi@mail.com	08123456789
	2	Rayhan	rayhan@mail.com	08123456780
	3	Agyl	agyl@mail.com	08123456781
	4	Aulia	aulia@mail.com	08123456782
	5	Dimas	dimas@mail.com	08123456783
	6	Edo	edo@mail.com	08123456784
	7	Salah	salah@mail.com	08123456785
	8	Juan	juan@mail.com	08123456786
	9	Memei	memei@mail.com	08123456787
	10	Rahman	rahman@mail.com	08123456788
•	NULL	NULL	NULL	NULL

Gambar 10. Tabel Pengguna

	Id_Pengeluaran	Id_Pengguna	Tanggal	Jumlah
▶	1	1	2025-05-01	500000.00
	2	2	2025-05-02	400000.00
	3	3	2025-05-03	700000.00
	4	4	2025-05-04	600000.00
	5	5	2025-05-05	300000.00
	6	6	2025-05-06	200000.00
	7	7	2025-05-07	100000.00
	8	8	2025-05-08	900000.00
	9	9	2025-05-09	850000.00
	10	10	2025-05-10	750000.00
•	NULL	NULL	NULL	NULL

Gambar 11. Tabel Pengeluaran

	Id_Pemasukan	Id_Pengguna	Tanggal	Jumlah
▶	1	1	2025-05-02	3000000.00
	2	2	2025-05-02	2500000.00
	3	3	2025-05-02	2000000.00
	4	4	2025-05-02	1800000.00
	5	5	2025-05-02	2200000.00
	6	6	2025-05-02	2700000.00
	7	7	2025-05-02	1500000.00
	8	8	2025-05-02	1900000.00
	9	9	2025-05-02	3100000.00
	10	10	2025-05-02	3300000.00
•	NULL	NULL	NULL	NULL

Gambar 12. Tabel Pemasukan

	Id_Hutang	Id_Pengguna	Tanggal_Hutang	Jatuh_Tempo	Jumlah
▶	1	1	2025-05-03	2025-06-01	1000000.00
	2	2	2025-05-03	2025-06-01	500000.00
	3	3	2025-05-03	2025-06-01	750000.00
	4	4	2025-05-03	2025-06-01	650000.00
	5	5	2025-05-03	2025-06-01	900000.00
	6	6	2025-05-03	2025-06-01	300000.00
	7	7	2025-05-03	2025-06-01	450000.00
	8	8	2025-05-03	2025-06-01	800000.00
	9	9	2025-05-03	2025-06-01	700000.00
	10	10	2025-05-03	2025-06-01	600000.00
•	NULL	NULL	NULL	NULL	NULL

Gambar 13. Tabel Hutang

	Id_Tabungan	Id_Pengguna	Jumlah	Target
▶	1	1	1500000.00	5000000.00
	2	2	1400000.00	4000000.00
	3	3	1300000.00	3000000.00
	4	4	1600000.00	3500000.00
	5	5	1700000.00	3700000.00
	6	6	1800000.00	3600000.00
	7	7	1200000.00	3200000.00
	8	8	1100000.00	3300000.00
	9	9	1900000.00	3900000.00
	10	10	2000000.00	4000000.00
•	NULL	NULL	NULL	NULL

Gambar 14. Tabel Tabungan

	Id_Asset	Id_Pengguna	Jenis	Nilai
▶	1	1	Motor	12000000.00
	2	2	Laptop	8000000.00
	3	3	Handphone	4000000.00
	4	4	Sepeda	3000000.00
	5	5	TV	3500000.00
	6	6	PC	7000000.00
	7	7	Tablet	4500000.00
	8	8	Camera	6000000.00
	9	9	Jam Tangan	2000000.00
	10	10	Kulkas	5000000.00
•	NULL	NULL	NULL	NULL

Gambar 15. Tabel Asset

	Id_Laporan	Id_Pengguna	Kategori	Tanggal
▶	1	1	Bulanan	2025-05-04
	2	2	Bulanan	2025-05-04
	3	3	Bulanan	2025-05-04
	4	4	Bulanan	2025-05-04
	5	5	Bulanan	2025-05-04
	6	6	Bulanan	2025-05-04
	7	7	Bulanan	2025-05-04
	8	8	Bulanan	2025-05-04
	9	9	Bulanan	2025-05-04
	10	10	Bulanan	2025-05-04
•	NULL	NULL	NULL	NULL

Gambar 16. Tabel Laporan

	Id_Pengeluaran	Id_Laporan
▶ 1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
10	10	

Gambar 17. Tabel Mencatat Pengeluaran

	Id_Pemasukan	Id_Laporan
▶ 1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
10	10	

Gambar 18. Tabel Mencatat Pemasukan

	Id_Hutang	Id_Laporan
▶ 1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
10	10	

Gambar 19. Tabel Mencatat Hutang

	Id_Asset	Id_Laporan
▶ 1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
10	10	

Gambar 20. Tabel Mencatat Asset

	Id_Tabungan	Id_Laporan
▶	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	10	10

Gambar 21. Tabel Mencatat Tabungan

3. Alias dan Operator

Printscreen Source Code :

```

2 • SELECT p.Nama, pem.Jumlah AS Pemasukan, peng.Jumlah AS Pengeluaran
3 FROM Pengguna p
4 JOIN Pemasukan pem ON p.Id_Pengguna = pem.Id_Pengguna
5 JOIN Pengeluaran peng ON p.Id_Pengguna = peng.Id_Pengguna
6 WHERE pem.Jumlah > 1000000;
7

```

Gambar 22. Alias dan Operator

Source Code :

```

SELECT p.Nama, pem.Jumlah AS Pemasukan, peng.Jumlah AS Pengeluaran
FROM Pengguna p
JOIN Pemasukan pem ON p.Id_Pengguna = pem.Id_Pengguna
JOIN Pengeluaran peng ON p.Id_Pengguna = peng.Id_Pengguna
WHERE pem.Jumlah > 1000000;

```

Penjelasan Source Code :

Query SQL tersebut bertujuan untuk menampilkan data nama pengguna beserta jumlah pemasukan dan pengeluaran mereka, namun hanya untuk pengguna yang memiliki jumlah pemasukan lebih dari 1.000.000. Proses ini dilakukan dengan menggabungkan tiga tabel utama, yaitu Pengguna, Pemasukan, dan Pengeluaran.

Pertama, tabel Pengguna (yang diberi alias p) digunakan sebagai basis untuk mengambil data nama pengguna. Kemudian, dilakukan operasi JOIN dengan tabel Pemasukan (dengan alias pem) melalui kolom Id_Pengguna, yang merupakan kunci relasi antar tabel, sehingga data pemasukan yang sesuai untuk setiap pengguna dapat diambil. Selanjutnya, dilakukan lagi JOIN dengan tabel Pengeluaran (beralias

peng) juga melalui kolom Id_Pengguna, agar data pengeluaran masing-masing pengguna ikut ditampilkan.

Hasil akhir dari query ini akan berupa daftar nama pengguna dengan kolom pemasukan dan pengeluaran yang dimiliki masing-masing, tetapi hanya jika pemasukan mereka melebihi 1.000.000.

Output :

	Nama	Pemasukan	Pengeluaran
▶	Ghazi	3000000.00	500000.00
	Rayhan	2500000.00	400000.00
	Agyl	2000000.00	700000.00
	Aulia	1800000.00	600000.00
	Dimas	2200000.00	300000.00
	Edo	2700000.00	200000.00
	Salah	1500000.00	100000.00
	Juan	1900000.00	900000.00
	Memei	3100000.00	850000.00
	Rahman	3300000.00	750000.00

Gambar 23. Tabel Output Alias dan Operator

4. Fungsi Join

Printscreen Source Code :

```

8 • SELECT p>Nama, pe.Tanggal, pe.Jumlah
9 FROM Pengguna p
10 INNER JOIN Pengeluaran pe ON p.Id_Pengguna = pe.Id_Pengguna;
11
12 • SELECT p>Nama, t.Jumlah AS Jumlah_Tabungan
13 FROM Pengguna p
14 LEFT JOIN Tabungan t ON p.Id_Pengguna = t.Id_Pengguna;
15
16 • SELECT l.Kategori, pm.Jumlah AS Jumlah_Pemasukan
17 FROM Laporan l
18 RIGHT JOIN Catat_Pemasukan cp ON l.Id_Laporan = cp.Id_Laporan
19 RIGHT JOIN Pemasukan pm ON cp.Id_Pemasukan = pm.Id_Pemasukan;
20
21 • SELECT p>Nama, h.Jumlah AS Jumlah_Hutang
22 FROM Pengguna p
23 LEFT JOIN Hutang h ON p.Id_Pengguna = h.Id_Pengguna
24 UNION
25 SELECT p>Nama, h.Jumlah
26 FROM Pengguna p
27 RIGHT JOIN Hutang h ON p.Id_Pengguna = h.Id_Pengguna;
28
29 • SELECT p>Nama, l.Kategori
30 FROM Pengguna p
31 CROSS JOIN (SELECT DISTINCT Kategori FROM Laporan) l;

```

Gambar 24. Join

Source Code :

SELECT p>Nama, pe.Tanggal, pe.Jumlah

```

FROM Pengguna p
INNER JOIN Pengeluaran pe ON p.Id_Pengguna = pe.Id_Pengguna;
SELECT p.Nama, t.Jumlah AS Jumlah_Tabungan
FROM Pengguna p
LEFT JOIN Tabungan t ON p.Id_Pengguna = t.Id_Pengguna;

SELECT l.Kategori, pm.Jumlah AS Jumlah_Pemasukan
FROM Laporan l
RIGHT JOIN Catat_Pemasukan cp ON l.Id_Laporan = cp.Id_Laporan
RIGHT JOIN Pemasukan pm ON cp.Id_Pemasukan = pm.Id_Pemasukan;

SELECT p.Nama, h.Jumlah AS Jumlah_Hutang
FROM Pengguna p
LEFT JOIN Hutang h ON p.Id_Pengguna = h.Id_Pengguna
UNION
SELECT p.Nama, h.Jumlah
FROM Pengguna p
RIGHT JOIN Hutang h ON p.Id_Pengguna = h.Id_Pengguna;

SELECT p.Nama, l.Kategori
FROM Pengguna p
CROSS JOIN (SELECT DISTINCT Kategori FROM Laporan) l;

```

Penjelasan Source Code :

Query yang pertama ini mengambil nama pengguna (p.Nama), tanggal pengeluaran (pe.Tanggal), dan jumlah pengeluaran (pe.Jumlah) dari dua tabel, yaitu Pengguna dan Pengeluaran. INNER JOIN memastikan bahwa hanya data pengguna yang memiliki pengeluaran (yaitu, terdapat kecocokan Id_Pengguna di kedua tabel) yang ditampilkan. Jika ada pengguna tanpa data pengeluaran, maka mereka tidak akan muncul dalam hasil query ini.

Query kedua ini menampilkan semua nama pengguna beserta jumlah tabungan mereka. Dengan menggunakan LEFT JOIN, query akan tetap menampilkan semua pengguna, bahkan jika mereka tidak memiliki data tabungan di tabel Tabungan.

Jika tidak ada catatan tabungan untuk pengguna tertentu, maka kolom Jumlah_Tabungan akan bernilai NULL.

Query ketiga mengambil kategori laporan dan jumlah pemasukan. Data diambil dari tiga tabel, yaitu Laporan, Catat_Pemasukan, dan Pemasukan. RIGHT JOIN digunakan dua kali secara berturut-turut: pertama untuk menggabungkan Laporan dengan Catat_Pemasukan, lalu hasilnya digabung lagi dengan Pemasukan. Tujuan penggunaan RIGHT JOIN adalah agar semua data pemasukan tetap ditampilkan meskipun tidak memiliki relasi dengan laporan. Jika suatu pemasukan tidak terkait dengan laporan, maka kolom Kategori akan berisi NULL.

Query keempat ini menggabungkan dua query menggunakan UNION untuk mendapatkan hasil lengkap dari relasi antara pengguna dan hutangnya. Query pertama menggunakan LEFT JOIN, yang menampilkan semua pengguna termasuk yang tidak memiliki hutang (dengan nilai NULL). Query kedua menggunakan RIGHT JOIN, yang menampilkan semua hutang bahkan jika tidak ada data pengguna yang cocok. UNION menggabungkan hasil keduanya dan secara otomatis menghapus duplikat, sehingga semua pengguna dan semua data hutang yang mungkin tidak memiliki pasangan tetap ditampilkan.

Query terakhir menggunakan CROSS JOIN, yang menghasilkan produk kartesius antara tabel Pengguna dan daftar kategori laporan unik. Artinya, setiap pengguna akan dipasangkan dengan semua kategori laporan yang ada. Jika terdapat 5 pengguna dan 3 kategori unik, maka hasilnya akan berupa 15 baris. Ini berguna untuk menghasilkan kombinasi potensial dari data, seperti membuat daftar perencanaan atau template pengisian data untuk setiap kategori per pengguna.

Output :

	Nama	Tanggal	Jumlah
►	Ghazi	2025-05-01	500000.00
	Rayhan	2025-05-02	400000.00
	Agyl	2025-05-03	700000.00
	Aulia	2025-05-04	600000.00
	Dimas	2025-05-05	300000.00
	Edo	2025-05-06	200000.00
	Salah	2025-05-07	100000.00
	Juan	2025-05-08	900000.00
	Memei	2025-05-09	850000.00
	Rahman	2025-05-10	750000.00

Gambar 25. Tabel Output Inner Join

	Nama	Jumlah_Tabungan
►	Ghazi	1500000.00
	Rayhan	1400000.00
	Agyl	1300000.00
	Aulia	1600000.00
	Dimas	1700000.00
	Edo	1800000.00
	Salah	1200000.00
	Juan	1100000.00
	Memei	1900000.00
	Rahman	2000000.00

Gambar 26. Tabel Output Left Join

	Kategori	Jumlah_Pemasukan
►	Bulanan	3000000.00
	Bulanan	2500000.00
	Bulanan	2000000.00
	Bulanan	1800000.00
	Bulanan	2200000.00
	Bulanan	2700000.00
	Bulanan	1500000.00
	Bulanan	1900000.00
	Bulanan	3100000.00
	Bulanan	3300000.00

Gambar 27. Tabel Output Right Join

	Nama	Jumlah_Hutang
►	Ghazi	1000000.00
	Rayhan	500000.00
	Agyl	750000.00
	Aulia	650000.00
	Dimas	900000.00
	Edo	300000.00
	Salah	450000.00
	Juan	800000.00
	Memei	700000.00
	Rahman	600000.00

Gambar 28. Tabel Output Full Join

	Nama	Kategori
►	Rahman	Bulanan
	Memei	Bulanan
	Juan	Bulanan
	Salah	Bulanan
	Edo	Bulanan
	Dimas	Bulanan
	Aulia	Bulanan
	Agyl	Bulanan
	Rayhan	Bulanan
	Ghazi	Bulanan

Gambar 29. Tabel Output Cross Join

5. Function

Printscreen Source Code :

```
2 DELIMITER //
3 CREATE FUNCTION TotalTabungan(id INT) RETURNS DECIMAL(12,2)
4 DETERMINISTIC
5 BEGIN
6     DECLARE total DECIMAL(12,2);
7     SELECT SUM(Jumlah) INTO total FROM Tabungan WHERE Id_Pengguna = id;
8     RETURN IFNULL(total, 0.00);
9 END;
10 //
11 DELIMITER ;
```

Gambar 30. Function

Source Code :

```
DELIMITER //
CREATE FUNCTION TotalTabungan(id INT) RETURNS DECIMAL(12,2)
DETERMINISTIC
BEGIN
    DECLARE total DECIMAL(12,2);
    SELECT SUM(Jumlah) INTO total FROM Tabungan WHERE Id_Pengguna
= id;
    RETURN IFNULL(total, 0.00);
END;
//
DELIMITER ;
```

Penjelasan Source Code :

Kode di atas merupakan definisi sebuah fungsi bernama TotalTabungan dalam bahasa SQL pada sistem basis data MySQL. Fungsi ini dibuat menggunakan perintah **CREATE FUNCTION**, dan dirancang untuk menerima satu parameter berupa id bertipe INT, yang merepresentasikan ID dari seorang pengguna.

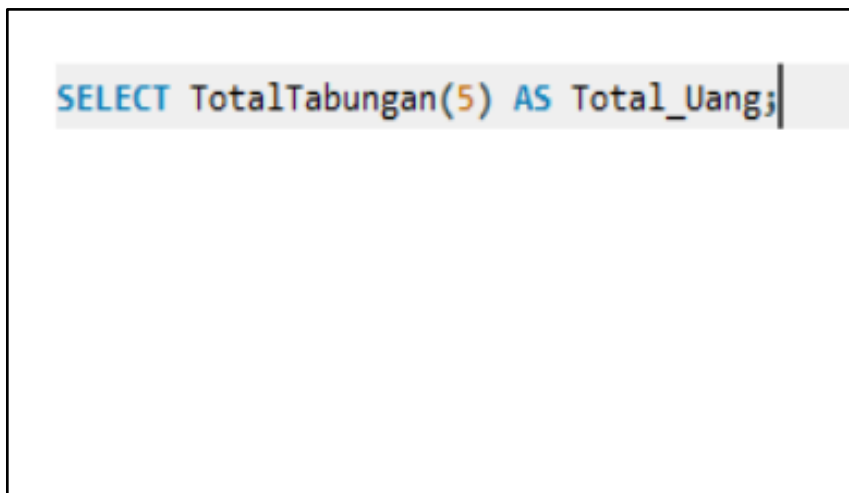
Fungsi ini akan mengembalikan nilai berupa angka desimal (**DECIMAL(12,2)**), yaitu total jumlah tabungan dari pengguna tersebut. Kata kunci **DETERMINISTIC** menyatakan bahwa fungsi ini akan selalu memberikan hasil yang sama jika

diberikan input yang sama, karena tidak bergantung pada faktor acak atau data eksternal lainnya.

Di dalam blok **BEGIN...END**, terdapat deklarasi variabel lokal bernama **total** yang digunakan untuk menyimpan hasil perhitungan jumlah (**SUM**) dari kolom Jumlah dalam tabel Tabungan, di mana nilai **Id_Pengguna** sesuai dengan parameter **id** yang diberikan.

Hasil perhitungan ini kemudian dikembalikan melalui pernyataan **RETURN**, dan jika hasil penjumlahan bernilai **NULL**, maka akan dikembalikan nilai default 0.00 menggunakan fungsi **IFNULL**. Delimiter diganti sementara menjadi **//** agar MySQL tidak mengakhiri perintah saat menemui tanda **;** di dalam blok fungsi, dan dikembalikan ke **;** setelah definisi fungsi selesai.

Printscreen Source Code :



Gambar 31. Penggunaan Function

Source Code :

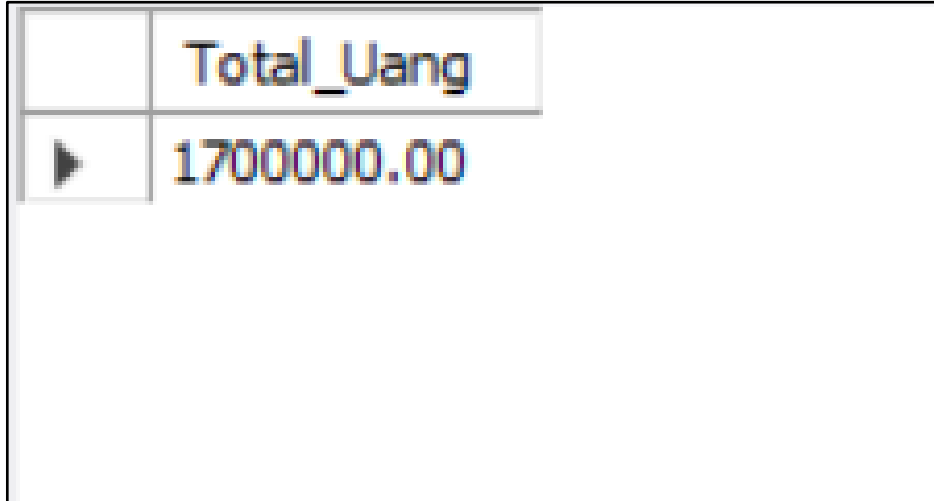
```
SELECT TotalTabungan(5) AS Total_Uang;
```

Penjelasan Source Code :

Kode **SELECT TotalTabungan(5) AS Total_Uang;** digunakan untuk menjalankan fungsi **TotalTabungan** yang sebelumnya telah didefinisikan. Dalam perintah ini, angka 5 diberikan sebagai argumen ke dalam fungsi, yang berarti sistem akan menghitung total jumlah tabungan untuk pengguna dengan **Id_Pengguna** bernilai 5 dari tabel Tabungan. Fungsi **TotalTabungan** akan menjumlahkan seluruh nilai pada kolom Jumlah yang berkaitan dengan ID tersebut, dan jika tidak ditemukan data tabungan, maka akan mengembalikan nilai 0.00

secara default. Hasil dari fungsi ini kemudian ditampilkan dalam bentuk sebuah kolom bernama Total_Uang, sebagaimana ditentukan oleh alias.

Output :

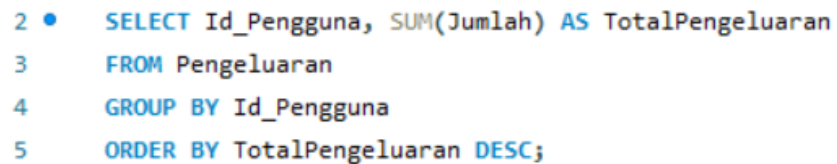


Total_Uang
1700000.00

Gambar 32. Tabel Output Function

6. Grouping dan Sorting

Printscreen Source Code :



```
2 • SELECT Id_Pengguna, SUM(Jumlah) AS TotalPengeluaran
3 FROM Pengeluaran
4 GROUP BY Id_Pengguna
5 ORDER BY TotalPengeluaran DESC;
```

Gambar 33. Grouping dan Sorting

Source Code :

```
SELECT Id_Pengguna, SUM(Jumlah) AS TotalPengeluaran
FROM Pengeluaran
GROUP BY Id_Pengguna
ORDER BY TotalPengeluaran DESC;
```

Penjelasan Source Code :

Kode `SELECT Id_Pengguna, SUM(Jumlah) AS TotalPengeluaran FROM Pengeluaran GROUP BY Id_Pengguna ORDER BY TotalPengeluaran DESC;` digunakan untuk menghasilkan daftar total pengeluaran yang dilakukan oleh setiap pengguna yang tercatat dalam tabel Pengeluaran. Perintah `SELECT Id_Pengguna, SUM(Jumlah) AS TotalPengeluaran` berarti sistem akan menampilkan dua kolom,

kolom pertama adalah Id_Pengguna, yaitu identitas unik dari masing-masing pengguna, dan kolom kedua adalah hasil penjumlahan dari seluruh nilai Jumlah (yang merepresentasikan besarnya pengeluaran) untuk setiap pengguna, yang ditampilkan dengan nama alias TotalPengeluaran.

Untuk memastikan bahwa jumlah pengeluaran dihitung per pengguna, digunakan klausa **GROUP BY Id_Pengguna**, yang mengelompokkan data berdasarkan nilai Id_Pengguna. Dengan demikian, setiap baris hasil query mewakili satu pengguna dengan total pengeluarannya masing-masing. Selanjutnya, hasil tersebut diurutkan berdasarkan nilai TotalPengeluaran secara menurun (descending), seperti yang ditentukan oleh klausa **ORDER BY TotalPengeluaran DESC**, sehingga pengguna dengan total pengeluaran terbesar akan ditampilkan di bagian atas hasil query

Output :

	Id_Pengguna	TotalPengeluaran
▶	8	900000.00
	9	850000.00
	10	750000.00
	3	700000.00
	4	600000.00
	1	500000.00
	2	400000.00
	5	300000.00
	6	200000.00
	7	100000.00

Gambar 34. Tabel Output Grouping dan Sorting

7. View

Printscreen Source Code :

```

2 • CREATE VIEW View_Laporan_Kuangan AS
3 SELECT p.Nama, l.Kategori, l.Tanggal
4 FROM Laporan l
5 JOIN Pengguna p ON l.Id_Pengguna = p.Id_Pengguna;

```

Gambar 35. View

Source Code :

```
CREATE VIEW View_Laporan_Kuangan AS  
SELECT p.Nama, l.Kategori, l.Tanggal  
FROM Laporan l  
JOIN Pengguna p ON l.Id_Pengguna = p.Id_Pengguna;
```

Penjelasan Source Code :

Kode `CREATE VIEW View_Laporan_Kuangan AS SELECT p.Nama, l.Kategori, l.Tanggal FROM Laporan l JOIN Pengguna p ON l.Id_Pengguna = p.Id_Pengguna;` digunakan untuk membuat sebuah view atau tampilan virtual di dalam basis data dengan nama `View_Laporan_Kuangan`. View ini merupakan hasil dari query `SELECT` yang mengambil dan menggabungkan data dari dua tabel, yaitu `Laporan` dan `Pengguna`, melalui operasi `JOIN`.

Dalam hal ini, tabel `Laporan` diberi alias `l` dan tabel `Pengguna` diberi alias `p`. Keduanya digabungkan dengan kondisi `ON l.Id_Pengguna = p.Id_Pengguna`, yang berarti baris-baris dari tabel `Laporan` akan dicocokkan dengan baris-baris dari tabel `Pengguna` yang memiliki nilai `Id_Pengguna` yang sama. Setelah digabungkan, query memilih tiga kolom untuk ditampilkan dalam view, yaitu `Nama` dari tabel `Pengguna`, serta `Kategori` dan `Tanggal` dari tabel `Laporan`.

Printscreen Source Code :

Gambar 36. Memanggil View

Source Code :

```
SELECT * FROM View_Laporan_Kuangan;
```

Penjelasan Source Code :

Perintah `SELECT * FROM View_Laporan_Kuangan;` digunakan untuk mengambil dan menampilkan seluruh data yang terdapat di dalam view bernama

View_Laporan_Keuangan. Tanda asterisk (*) menunjukkan bahwa semua kolom yang tersedia dalam view akan ditampilkan tanpa perlu menyebutkan nama kolom satu per satu.

Karena View_Laporan_Keuangan sebelumnya didefinisikan sebagai gabungan antara tabel Laporan dan Pengguna, maka setiap baris yang ditampilkan oleh perintah ini akan berisi nama pengguna (Nama), kategori laporan (Kategori), dan tanggal laporan (Tanggal).

8. Stored Procedure

Printscreen Source Code :

```
2 DELIMITER //
3 CREATE PROCEDURE sp_TambahPengguna(
4     IN p_nama VARCHAR(100),
5     IN p_email VARCHAR(100),
6     IN p_no_hp VARCHAR(20)
7 )
8 BEGIN
9     DECLARE new_id INT;
10
11     SELECT IFNULL(MAX(Id_Pengguna), 0) + 1 INTO new_id FROM Pengguna;
12
13     INSERT INTO Pengguna (Id_Pengguna, Nama, Email, No_Handphone)
14     VALUES (new_id, p_nama, p_email, p_no_hp);
15
16     INSERT INTO Tabungan (Id_Tabungan, Id_Pengguna, Jumlah, Target)
17     VALUES (new_id, new_id, 0, 0);
18 END //
19 DELIMITER ;
```

Gambar 37. Stored Procedure

Source Code :

DELIMITER //

CREATE PROCEDURE sp_TambahPengguna(

IN p_nama VARCHAR(100),

IN p_email VARCHAR(100),

IN p_no_hp VARCHAR(20)

)

BEGIN

DECLARE new_id INT;

SELECT IFNULL(MAX(Id_Pengguna), 0) + 1 INTO new_id FROM
Pengguna;

INSERT INTO Pengguna (Id_Pengguna, Nama, Email, No_Handphone)

```
VALUES (new_id, p_nama, p_email, p_no_hp);
```

```
INSERT INTO Tabungan (Id_Tabungan, Id_Pengguna, Jumlah, Target)
```

```
VALUES (new_id, new_id, 0, 0);
```

```
END //
```

```
DELIMITER ;
```

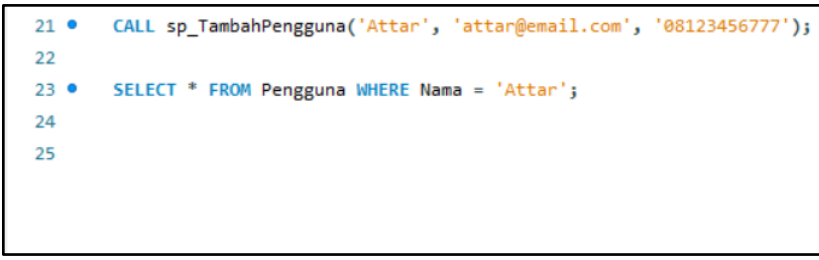
Penjelasan Source Code :

Kode tersebut adalah definisi dari sebuah stored procedure bernama `sp_TambahPengguna`, yang dirancang untuk menambahkan data pengguna baru ke dalam sistem secara otomatis sekaligus membuat catatan tabungan default untuk pengguna tersebut. Stored procedure ini menerima tiga parameter input, yaitu `p_nama` (nama pengguna), `p_email` (alamat email), dan `p_no_hp` (nomor handphone).

Di dalam prosedur, langkah pertama adalah mendeklarasikan sebuah variabel lokal bernama `new_id` bertipe `INT` yang akan digunakan untuk menyimpan ID pengguna baru. Nilai dari `new_id` diambil dengan cara mencari nilai maksimum dari kolom `Id_Pengguna` di tabel `Pengguna`, kemudian ditambahkan satu. Jika tidak ada data, maka akan menghasilkan 0 dan `new_id` akan menjadi 1. Ini berfungsi sebagai cara manual untuk membuat auto-increment ID secara eksplisit.

Setelah nilai `new_id` didapat, baris baru akan disisipkan ke dalam tabel `Pengguna` menggunakan nilai `new_id` tersebut sebagai `Id_Pengguna`, dan diikuti dengan nama, email, serta nomor handphone yang diberikan melalui parameter input. Kemudian, prosedur melanjutkan dengan memasukkan data ke tabel `Tabungan`, di mana akan dibuat tabungan default untuk pengguna baru tersebut. Nilai `Id_Tabungan` dan `Id_Pengguna` pada tabel `Tabungan` akan sama dengan `new_id`, sedangkan kolom `Jumlah` dan `Target` akan diisi dengan nilai 0.

Printscreen Source Code :



```
21 • CALL sp_TambahPengguna('Attar', 'attar@email.com', '08123456777');  
22  
23 • SELECT * FROM Pengguna WHERE Nama = 'Attar';  
24  
25
```

Gambar 38. Penggunaan Stored Procedure

Source Code :

```
CALL sp_TambahPegguna('Attar', 'attar@email.com', '08123456777');
```

```
SELECT * FROM Pengguna WHERE Nama = 'Attar';
```

Penjelasan Source Code :

Kode tersebut terdiri dari dua perintah SQL yang digunakan untuk menambahkan data pengguna baru ke dalam sistem dan kemudian menampilkan data pengguna tersebut setelah ditambahkan. Perintah pertama, **CALL sp_TambahPegguna('Attar', 'attar@email.com', '08123456777');**, memanggil stored procedure **sp_TambahPegguna** yang sebelumnya telah didefinisikan. Stored procedure ini secara otomatis akan menambahkan pengguna baru bernama Attar dengan email attar@email.com dan nomor handphone 08123456777 ke dalam tabel Pengguna. Selain itu, prosedur ini juga akan membuatkan entri tabungan awal untuk pengguna tersebut di tabel Tabungan dengan nilai saldo (Jumlah) dan target tabungan (Target) bernilai nol.

Setelah prosedur dijalankan dan data berhasil dimasukkan, perintah kedua, yaitu **SELECT * FROM Pengguna WHERE Nama = 'Attar';**, digunakan untuk mengambil dan menampilkan seluruh data dari tabel Pengguna yang memiliki nilai Nama sama dengan 'Attar'. Tujuannya adalah untuk memverifikasi bahwa data pengguna benar-benar telah ditambahkan ke dalam basis data. Output dari query ini akan menunjukkan informasi lengkap pengguna tersebut, termasuk Id_Pengguna yang telah di-generate secara otomatis dalam prosedur, serta email dan nomor handphone yang telah diberikan.

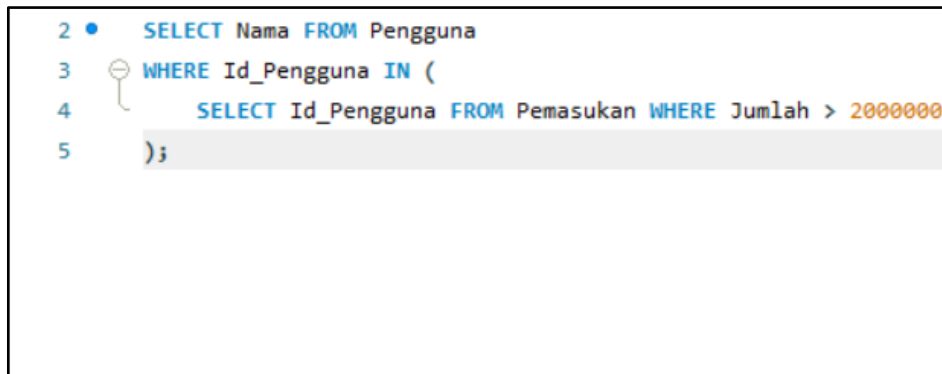
Output :

	Id_Pengguna	Nama	Email	No_Handphone
▶	11	Attar	attar@email.com	08123456777
*	NULL	NULL	NULL	NULL

Gambar 39. Tabel Output Stored Procedure

9. Nested Query

Printscreen Source Code :



```
2 • SELECT Nama FROM Pengguna
3 WHERE Id_Pengguna IN (
4     SELECT Id_Pengguna FROM Pemasukan WHERE Jumlah > 2000000
5 );
```

Gambar 40. Nested Query

Source Code :

```
SELECT Nama FROM Pengguna
WHERE Id_Pengguna IN (
    SELECT Id_Pengguna FROM Pemasukan WHERE Jumlah > 2000000
);
```

Penjelasan Source Code :

Kode SQL ini merupakan sebuah query yang bertujuan untuk menampilkan nama-nama pengguna yang memiliki pemasukan lebih dari 2.000.000.

Pada bagian subquery `SELECT Id_Pengguna FROM Pemasukan WHERE Jumlah > 2000000`, sistem akan mencari semua ID pengguna yang memiliki record pemasukan dengan jumlah lebih besar dari 2.000.000 di tabel Pemasukan. Hasil dari subquery ini berupa daftar ID pengguna yang memenuhi kriteria tersebut.

Kemudian, query utama `SELECT Nama FROM Pengguna WHERE Id_Pengguna IN (...)` akan mengambil data nama dari tabel Pengguna dimana ID pengguna tersebut termasuk dalam daftar ID yang dihasilkan oleh subquery.

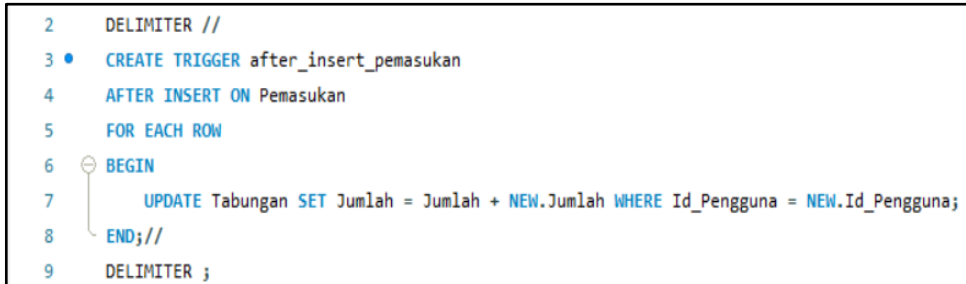
Output :

	Nama
▶	Ghazi
	Rayhan
	Dimas
	Edo
	Memei
	Rahman

Gambar 41. Tabel Output Nested Query

10. Trigger

Printscreen Source Code :

A screenshot of a SQL code editor showing the creation of a trigger. The code is as follows:

```
2 DELIMITER //  
3 CREATE TRIGGER after_insert_pemasukan  
4 AFTER INSERT ON Pemasukan  
5 FOR EACH ROW  
6 BEGIN  
7     UPDATE Tabungan SET Jumlah = Jumlah + NEW.Jumlah WHERE Id_Pengguna = NEW.Id_Pengguna;  
8 END;//  
9 DELIMITER ;
```

Gambar 42. Trigger

Source Code :

```
DELIMITER //  
CREATE TRIGGER after_insert_pemasukan  
AFTER INSERT ON Pemasukan  
FOR EACH ROW  
BEGIN  
    UPDATE Tabungan SET Jumlah = Jumlah + NEW.Jumlah WHERE  
Id_Pengguna = NEW.Id_Pengguna;  
END;//  
DELIMITER ;
```

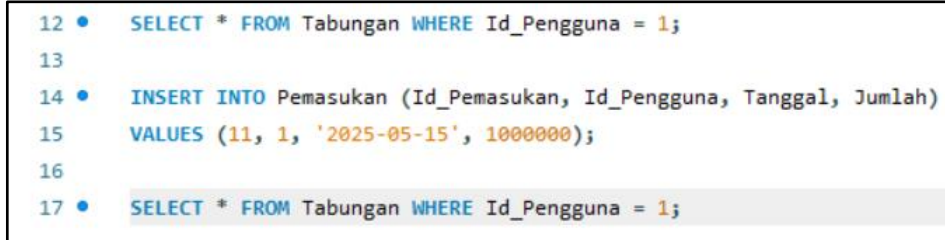
Penjelasan Source Code :

Trigger `after_insert_pemasukan` adalah sebuah mekanisme otomatis di dalam database MySQL yang dirancang untuk menjaga konsistensi data antara tabel Pemasukan dan tabel Tabungan. Trigger ini akan dijalankan secara otomatis setiap kali ada baris baru (record) yang dimasukkan ke dalam tabel Pemasukan, dan dieksekusi setelah proses **INSERT** terjadi, sesuai dengan deklarasi **AFTER INSERT ON Pemasukan**. **FOR EACH ROW** menunjukkan bahwa trigger ini akan bekerja untuk setiap baris yang dimasukkan, bukan hanya sekali untuk seluruh operasi insert, yang berarti sangat berguna jika ada lebih dari satu pemasukan ditambahkan dalam satu query.

Di dalam blok **BEGIN...END**, terdapat perintah SQL yang akan memperbarui nilai Jumlah pada tabel Tabungan. Secara spesifik, baris **UPDATE Tabungan SET Jumlah = Jumlah + NEW.Jumlah WHERE Id_Pengguna = NEW.Id_Pengguna;** berfungsi untuk menambahkan jumlah pemasukan ke saldo tabungan milik

pengguna terkait. Di sini, **NEW** adalah keyword yang merepresentasikan baris data yang baru saja dimasukkan ke tabel Pemasukan.

Printscreen Source Code :



```
12 • SELECT * FROM Tabungan WHERE Id_Pengguna = 1;  
13  
14 • INSERT INTO Pemasukan (Id_Pemasukan, Id_Pengguna, Tanggal, Jumlah)  
15 VALUES (11, 1, '2025-05-15', 1000000);  
16  
17 • SELECT * FROM Tabungan WHERE Id_Pengguna = 1;
```

Gambar 43. Penggunaan Trigger

Source Code :

SELECT * FROM Tabungan WHERE Id_Pengguna = 1;

INSERT INTO Pemasukan (Id_Pemasukan, Id_Pengguna, Tanggal, Jumlah)
VALUES (11, 1, '2025-05-15', 1000000);

SELECT * FROM Tabungan WHERE Id_Pengguna = 1;

Penjelasan Source Code :

Pertama, perintah **SELECT * FROM Tabungan WHERE Id_Pengguna = 1;** digunakan untuk menampilkan data tabungan pengguna dengan ID 1 sebelum adanya pemasukan baru. Ini memberikan gambaran tentang saldo awal pengguna tersebut.

Kemudian, perintah **INSERT INTO Pemasukan** menambahkan record baru ke tabel Pemasukan dengan nilai ID_Pemasukan 11, ID_Pengguna 1, tanggal 15 Mei 2025, dan jumlah sebesar 1.000.000. Pada saat inilah **trigger after_insert_pemasukan** secara otomatis diaktifkan karena ada data baru yang dimasukkan ke tabel Pemasukan. Trigger tersebut akan mengeksekusi perintah **UPDATE** untuk menambahkan nilai Jumlah pemasukan baru ke saldo tabungan pengguna dengan ID 1 di tabel Tabungan.

Terakhir, perintah **SELECT * FROM Tabungan WHERE Id_Pengguna = 1;** yang kedua dijalankan untuk menampilkan data tabungan pengguna dengan ID 1 setelah pemasukan baru dan operasi trigger. Hasilnya akan menunjukkan bahwa saldo tabungan pengguna tersebut telah bertambah sebesar 1.000.000 dari saldo awal, membuktikan bahwa trigger telah bekerja sebagaimana mestinya.

Output :

	Id_Tabungan	Id_Pengguna	Jumlah	Target
▶	1	1	1500000.00	5000000.00
✱	NULL	NULL	NULL	NULL

Gambar 44. Output Sebelum Dilakukan Insert

	Id_Tabungan	Id_Pengguna	Jumlah	Target
▶	1	1	2500000.00	5000000.00
✱	NULL	NULL	NULL	NULL

Gambar 45. Output Setelah Dilakukan Insert

11. DCL (Data Control Language)

Printscreen Source Code :

```

2 • CREATE USER 'app_user'@'localhost' IDENTIFIED BY 'user123_';
3
4 • GRANT SELECT, INSERT, UPDATE ON financial_management.Pengguna TO 'app_user'@'localhost';
5 • GRANT SELECT, INSERT, UPDATE ON financial_management.Pemasukan TO 'app_user'@'localhost';
6 • GRANT SELECT, INSERT, UPDATE ON financial_management.Pengeluaran TO 'app_user'@'localhost';
7 • GRANT SELECT, INSERT, UPDATE ON financial_management.Hutang TO 'app_user'@'localhost';
8 • GRANT SELECT, INSERT, UPDATE ON financial_management.Tabungan TO 'app_user'@'localhost';
9 • GRANT SELECT, INSERT, UPDATE ON financial_management.Asset TO 'app_user'@'localhost';
10 • GRANT SELECT, INSERT, UPDATE ON financial_management.Laporan TO 'app_user'@'localhost';
11
12 • GRANT SELECT ON financial_management.View_Laporan_Keuangan TO 'app_user'@'localhost';
13 • GRANT EXECUTE ON FUNCTION financial_management.TotalTabungan TO 'app_user'@'localhost';
14 • GRANT EXECUTE ON PROCEDURE financial_management.InsertPemasukan TO 'app_user'@'localhost';
15
16 • CREATE USER 'admin_user'@'localhost' IDENTIFIED BY 'admin123_';
17
18 • GRANT ALL PRIVILEGES ON financial_management.* TO 'admin_user'@'localhost' WITH GRANT OPTION;
19
20 • FLUSH PRIVILEGES;

```

Gambar 46. DCL

Source Code :

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

**GRANT SELECT, INSERT, UPDATE ON financial_management.Pengguna
TO 'app_user'@'localhost';**

**GRANT SELECT, INSERT, UPDATE ON financial_management.Pemasukan
TO 'app_user'@'localhost';**


```
GRANT SELECT, INSERT, UPDATE ON financial_management.Pengeluaran  
TO 'app_user'@'localhost';
```

```
GRANT SELECT, INSERT, UPDATE ON financial_management.Hutang TO  
'app_user'@'localhost';
```

```
GRANT SELECT, INSERT, UPDATE ON financial_management.Tabungan  
TO 'app_user'@'localhost';
```

```
GRANT SELECT, INSERT, UPDATE ON financial_management.Asset TO  
'app_user'@'localhost';
```

```
GRANT SELECT, INSERT, UPDATE ON financial_management.Laporan TO  
'app_user'@'localhost';
```

```
GRANT SELECT ON financial_management.View_Laporan_Keuangan TO  
'app_user'@'localhost';
```

```
GRANT EXECUTE ON FUNCTION financial_management.TotalTabungan  
TO 'app_user'@'localhost';
```

```
GRANT EXECUTE ON PROCEDURE  
financial_management.InsertPemasukan TO 'app_user'@'localhost';
```

```
CREATE USER 'admin_user'@'localhost' IDENTIFIED BY 'admin123_';
```

```
GRANT ALL PRIVILEGES ON financial_management.* TO  
'admin_user'@'localhost' WITH GRANT OPTION;
```

```
FLUSH PRIVILEGES;
```

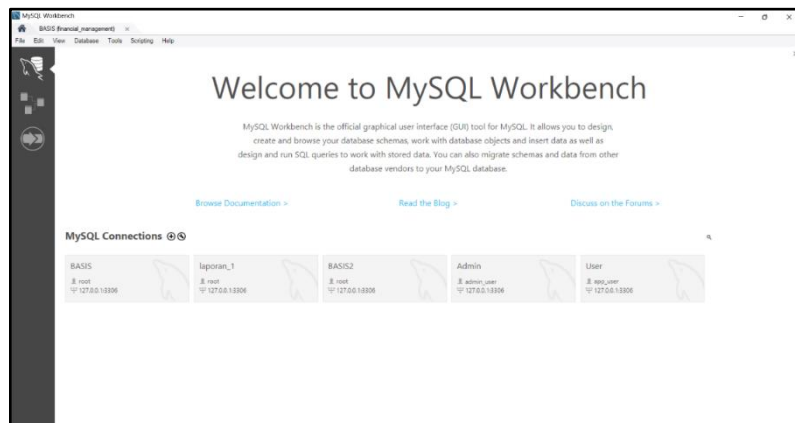
Penjelasan Source Code :

Kode SQL ini membangun sistem keamanan dan kontrol akses untuk database **financial_management** dengan membuat dua level pengguna yang berbeda. Pertama, dibuat user 'app_user' dengan password 'user123_' yang hanya dapat terhubung dari localhost. User ini diberikan hak akses terbatas berupa SELECT, INSERT, dan UPDATE pada tabel-tabel utama seperti Pengguna, Pemasukan, Pengeluaran, Hutang, Tabungan, Asset, dan Laporan. Selain itu, user aplikasi ini juga diberi hak untuk melihat view **View_Laporan_Keuangan** serta menjalankan

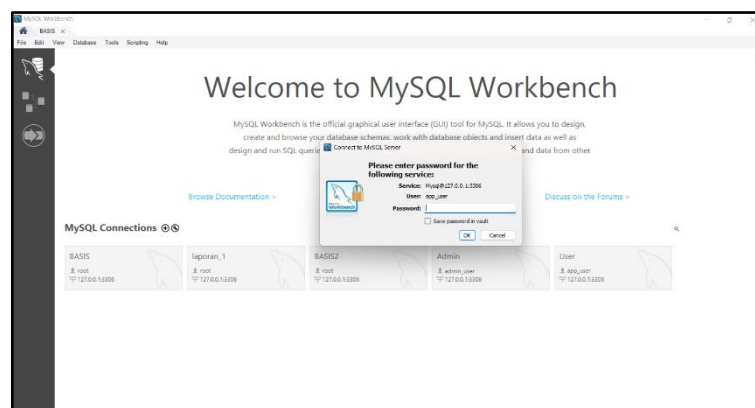
fungsi TotalTabungan dan stored procedure **InsertPemasukan**, tetapi tidak memiliki akses untuk menghapus data atau melakukan operasi DDL lainnya.

Kedua, dibuat user 'admin_user' dengan password 'admin123_' yang memiliki hak akses penuh. Berbeda dengan app_user, admin_user mendapatkan **ALL PRIVILEGES** pada seluruh objek dalam database financial_management, termasuk kemampuan untuk membuat, memodifikasi, dan menghapus tabel, view, prosedur, serta fungsi. Klausula **WITH GRANT OPTION** memberikan wewenang kepada admin_user untuk mendelegasikan hak aksesnya ke user lain. Perintah **FLUSH PRIVILEGES** di akhir berfungsi untuk memastikan semua perubahan hak akses yang telah ditetapkan segera berlaku tanpa perlu me-restart server database. Struktur ini menerapkan prinsip least privilege dimana app_user hanya mendapatkan akses minimal yang diperlukan untuk operasional aplikasi, sementara admin_user memiliki kendali penuh untuk keperluan administrasi dan maintenance database.

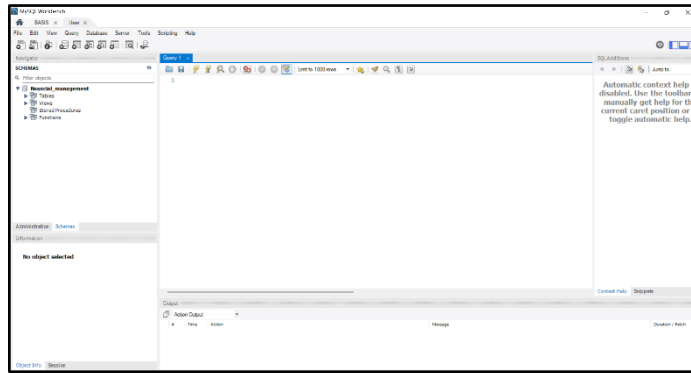
Output :



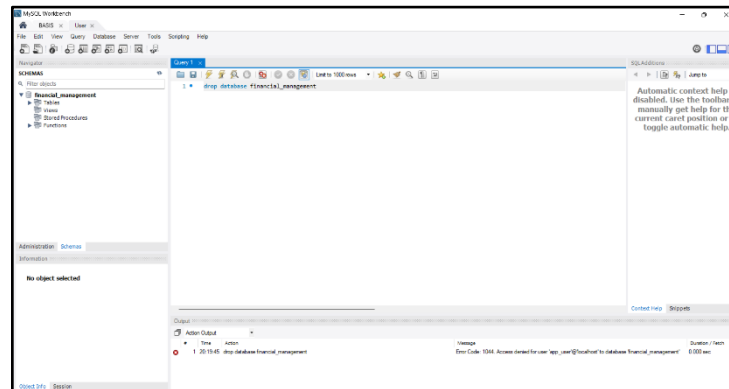
Gambar 47. Admin dan User



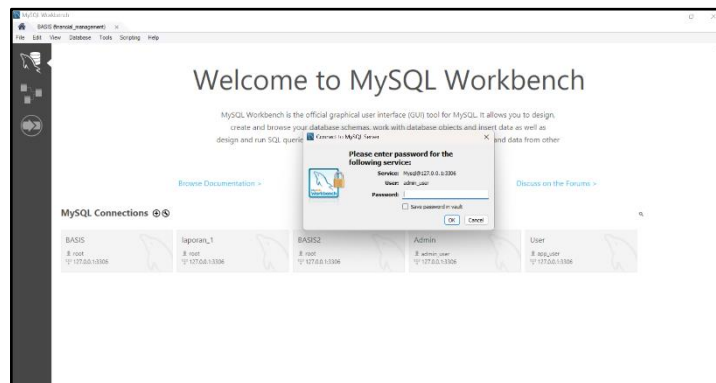
Gambar 48. User



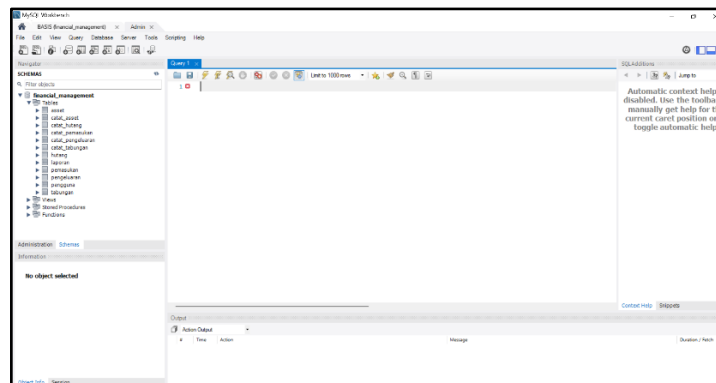
Gambar 49. Halaman User



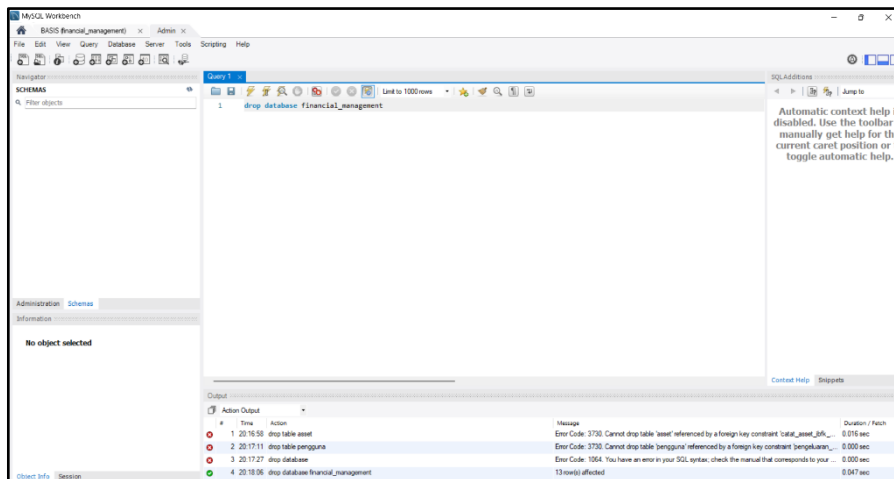
Gambar 50. User Mencoba Menghapus Database



Gambar 51. Admin



Gambar 52. Halaman Admin (1)



Gambar 53. Admin Berhasil Menghapus Database

Penjelasan Output :

Berdasarkan kedua gambar tersebut, dapat disimpulkan bahwa implementasi Data Control Language (DCL) dalam sistem manajemen database telah berhasil menciptakan struktur hak akses yang hierarkis dan aman. Pada Gambar 49. terlihat bahwa user secara sengaja dibatasi hak aksesnya. User hanya memiliki kemampuan untuk melakukan SELECT, INSERT, dan UPDATE pada tabel-tabel utama seperti Pengguna, Pemasukan, dan tabel penting lainnya. Selain itu, user juga diberikan akses terbatas untuk melihat **View_Laporan_Keuangan** serta menjalankan fungsi dan prosedur tertentu seperti TotalTabungan dan InsertPemasukan, namun secara eksplisit tidak memiliki hak untuk melakukan operasi DDL (Data Definition Language) seperti menghapus database atau memodifikasi struktur tabel.

Sebaliknya, Gambar 52. menunjukkan bahwa admin memiliki kendali penuh terhadap sistem. Admin diberi **ALL PRIVILEGES** yang mencakup seluruh objek dalam database **financial_management**, termasuk kemampuan untuk membuat, mengubah, dan menghapus tabel, view, stored procedure, serta fungsi. Yang lebih penting, admin juga memiliki **WITH GRANT OPTION** yang merupakan sebuah hak istimewa yang memungkinkannya untuk mendelegasikan wewenang kepada user lain. Perbedaan tingkat akses ini bukan tanpa alasan; admin membutuhkan fleksibilitas penuh untuk melakukan tugas-tugas administratif seperti maintenance database, optimasi performa, backup data, dan manajemen user, sementara di sisi lain harus memastikan bahwa user biasa tidak memiliki kemampuan yang bisa membahayakan integritas data.

KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan implementasi sistem manajemen keuangan pribadi yang telah dibangun, dapat disimpulkan bahwa struktur database yang dirancang telah memenuhi prinsip normalisasi dan relasional dengan baik.

Entity Relationship Diagram (ERD) yang dibuat menggambarkan hubungan antar entitas secara jelas, dengan Pengguna sebagai pusat yang terhubung ke berbagai transaksi keuangan seperti Pengeluaran, Pemasukan, Tabungan, Hutang, dan Aset, serta terintegrasi dengan Laporan sebagai entitas pelaporan. Penggunaan tabel penghubung memungkinkan fleksibilitas dalam mencatat transaksi ke dalam laporan, sementara penerapan constraints seperti PRIMARY KEY dan FOREIGN KEY menjaga integritas referensial data.

Fitur-fitur lanjutan seperti function, stored procedure, trigger, dan view memperkaya fungsionalitas sistem dengan otomatisasi dan kemudahan akses data. Selain itu, penerapan Data Control Language (DCL) yang membedakan hak akses antara user dan admin menunjukkan kesadaran akan keamanan data berdasarkan prinsip least privilege.

B. Saran

Untuk pengembangan lebih lanjut, sistem ini dapat ditingkatkan dengan beberapa penyempurnaan. Pertama, penambahan kolom "Kategori" pada tabel Pengeluaran dan Pemasukan akan memudahkan analisis pola keuangan pengguna. Kedua, implementasi transaksi pada stored procedure dan trigger akan memastikan konsistensi data saat operasi kompleks atau gagal di tengah proses. Ketiga, pembuatan indeks pada kolom yang sering digunakan dalam query dapat meningkatkan performa pencarian data. Keempat, perlu dipertimbangkan penambahan fitur notifikasi untuk mengingatkan pengguna tentang jatuh tempo hutang atau pencapaian target tabungan. Dari sisi keamanan, penggunaan password yang lebih kuat dan enkripsi data sensitif sangat disarankan. Terakhir, dokumentasi yang lebih rinci tentang alur kerja sistem dan pelatihan bagi pengguna akan memastikan pemanfaatan fitur secara optimal. Dengan penyempurnaan ini, sistem tidak hanya lebih robust secara teknis, tetapi juga lebih user-friendly dan aman dalam pengelolaan data keuangan pribadi.

DAFTAR PUSTAKA

- Costello, T., Blackshear, L., Costello, T., & Blackshear, L. (2020). Joins. Prepare Your Data for Tableau: A Practical Guide to the Tableau Data Prep Tool, 53-76.
- Dedianto, D. (2013). Sistem Trigger Database Pada SIAKAD Informatika. JUSTIN (Jurnal Sistem dan Teknologi Informasi), 1(1), 17-20.
- Effendi, D. (2018). 2 Menampilkan Data Menggunakan Perintah Sql Select.
- Małysiak-Mrozek, B., Mrozek, D., & Kozielski, S. (2009). Data grouping process in extended SQL language containing fuzzy elements. In Man-Machine Interactions (pp. 247-256). Springer Berlin Heidelberg.
- Naser Frak, A. (2016). Comparison study on sorting techniques in static data structure (Doctoral dissertation, Universiti Tun Hussein Onn Malaysia).
- Sequeda, J. F., Depena, R., & Miranker, D. P. (2009, September). Ultrawrap: Using sql views for rdb2rdf. In 8th International Semantic Web Conference (ISWC2009), Washington DC, USA.
- Putri, M. P., Nadeak, E., Malahayati, M., Rahmi, N., Rini, A., Sari, D. N., ... & Pratama, R. A. A. (2023). sistem manajemen basis data menggunakan MYSQL.
- Siregar, U. K., Sitakar, T. A., Haramain, S., Lubis, Z. N. S., Nadhirah, U., & Yahfizham, Y. (2024). Pengembangan database Management system menggunakan My SQL. Jurnal Sains, Teknologi & Komputer, 1(1), 8-12.

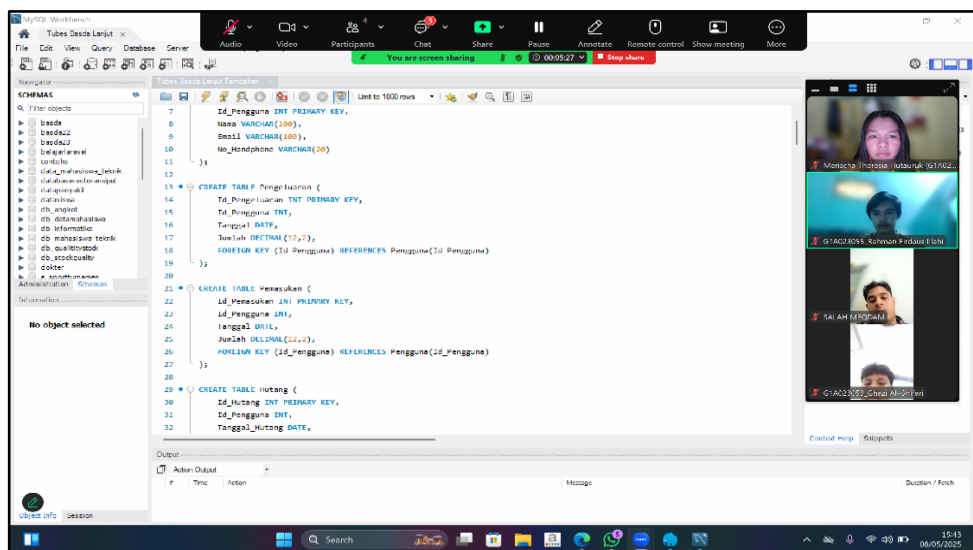
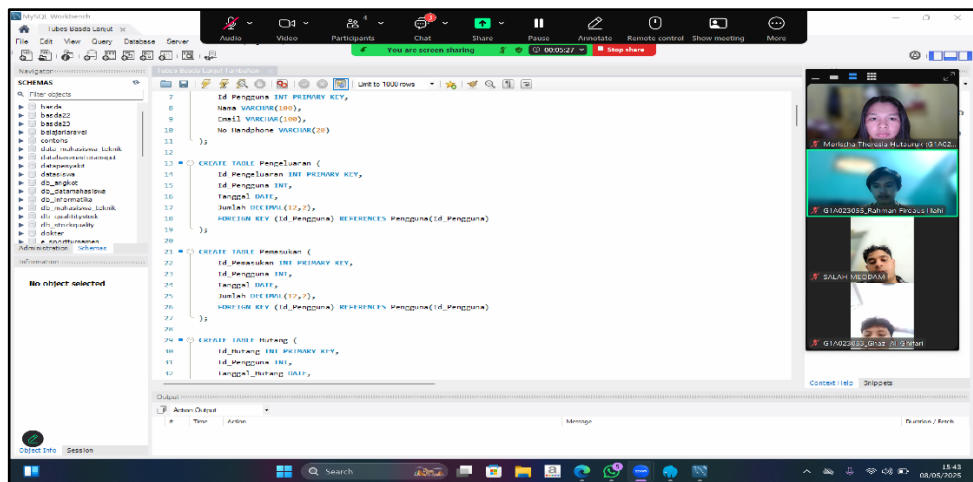
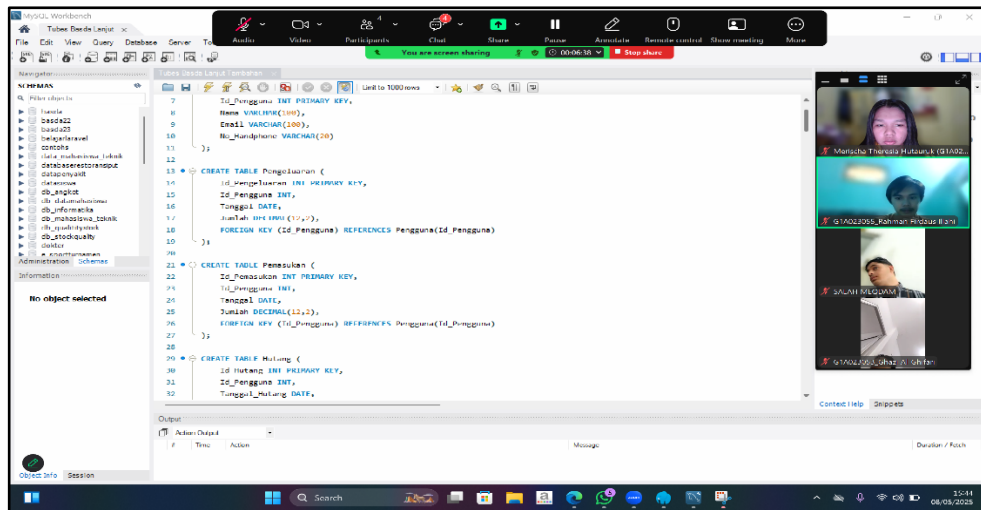
JOBDESK

Buat Laporan: Ghazi dan Salah

Source Code My Sql: Merischa

Diagram ERD : Rahman

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

PROYEK BASIS DATA LANJUT

Nama Mahasiswa	: 1. Ghazi Al-Ghifari	(G1A023053)
	: 2. Rahman Firdaus Illahi	(G1A023055)
	: 3. Merischa Theresia Hutauruk	(G1A023071)
	: 4. Salah Nasser Hasan Meqdam	(G1A023095)
Dosen	: 1. Dr. Endina Putri Purwandari, S. T., M.Kom	
	2. Ir. Tiara Eka Putri, S.T., M.Kom.	
Asisten Dosen	: 1. Merly Yuni Purnama	(G1A022006)
	2. Reksi Hendra Pratama	(G1A022032)
	3. Sinta Ezra Wati Gulo	(G1A022040)
	4. Fadlan Dwi Febrio	(G1A022051)
	5. Torang Four Yones Manullang	(G1A022052)
	6. Wahyu Ozorah Manurung	(G1A022060)

Laporan Tugas Besar	Catatan dan Tanda Tangan
Laporan Tugas Besar	

- | | |
|---------------------------|-------------|
| 7. Shalaudin Muhammad Sah | (G1A022070) |
| 8. Dian Ardiyanti Saputri | (G1A022084) |