

SISTEM BASIS DATA KLINIK
TUGAS BESAR MATA KULIAH SISTEM BASIS DATA
Dosen Pengampu : Faqih Hamami S.Kom., M.T



Disusun Oleh:

Gyebran Nauri Haikal (102022300389)

Lukas Ricky Krisjatmiko (102022330321)

Vanesa Rizka Alfatihah (102022300121)

Zaky Aprilian (102022300212)

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS REKAYASA INDUSTRI
UNIVERSITAS TELKOM
BANDUNG
2024

DAFTAR ISI

DAFTAR ISI.....	1
BAB I	
PENDAHULUAN.....	2
A. Latar Belakang.....	2
B. Tujuan dan Manfaat.....	2
BAB II	
PERANCANGAN SISTEM.....	4
A. Entity Relational Diagram (ERD).....	4
B. Relational Model.....	7
BAB III	
IMPLEMENTASI DASAR.....	9
A. Data Definition Language (DDL).....	9
B. Data Manipulation Language (DML).....	14
BAB IV	
IMPLEMENTASI LANJUTAN.....	21
A. Stored Procedure.....	21
B. Trigger.....	31
C. Data Control Language (DCL).....	42
BAB V	
LAINNYA.....	48
A. Pembagian Kelompok.....	48

BAB I

PENDAHULUAN

A. Latar Belakang

Di era digital ini, teknologi informasi sangat penting dalam berbagai sektor, termasuk kesehatan. Klinik membutuhkan sistem manajemen data yang efisien dan akurat untuk mengelola informasi pasien, jadwal dokter, stok obat, dan operasi lainnya. Saat ini, banyak klinik masih menggunakan pencatatan manual yang rentan terhadap kesalahan dan kehilangan data, menghambat pelayanan medis.

Sebagai mahasiswa teknologi informasi, saya melihat peluang untuk mengembangkan sistem database yang dapat meningkatkan efisiensi dan kualitas layanan klinik. Proyek ini bertujuan untuk merancang dan mengimplementasikan sistem database yang mampu mengelola data klinik secara efektif, mencakup pendaftaran pasien, rekam medis, jadwal dokter, dan inventaris obat.

Sistem ini juga akan dilengkapi dengan mekanisme keamanan untuk melindungi kerahasiaan data pasien. Dengan adanya sistem database ini, diharapkan klinik dapat meningkatkan efisiensi operasional, mengurangi kesalahan pencatatan, dan memberikan pelayanan yang lebih cepat dan akurat kepada pasien.

Melalui proyek ini, saya berharap dapat berkontribusi pada peningkatan manajemen data di klinik dan mendukung perbaikan layanan kesehatan di masyarakat.

B. Tujuan dan Manfaat

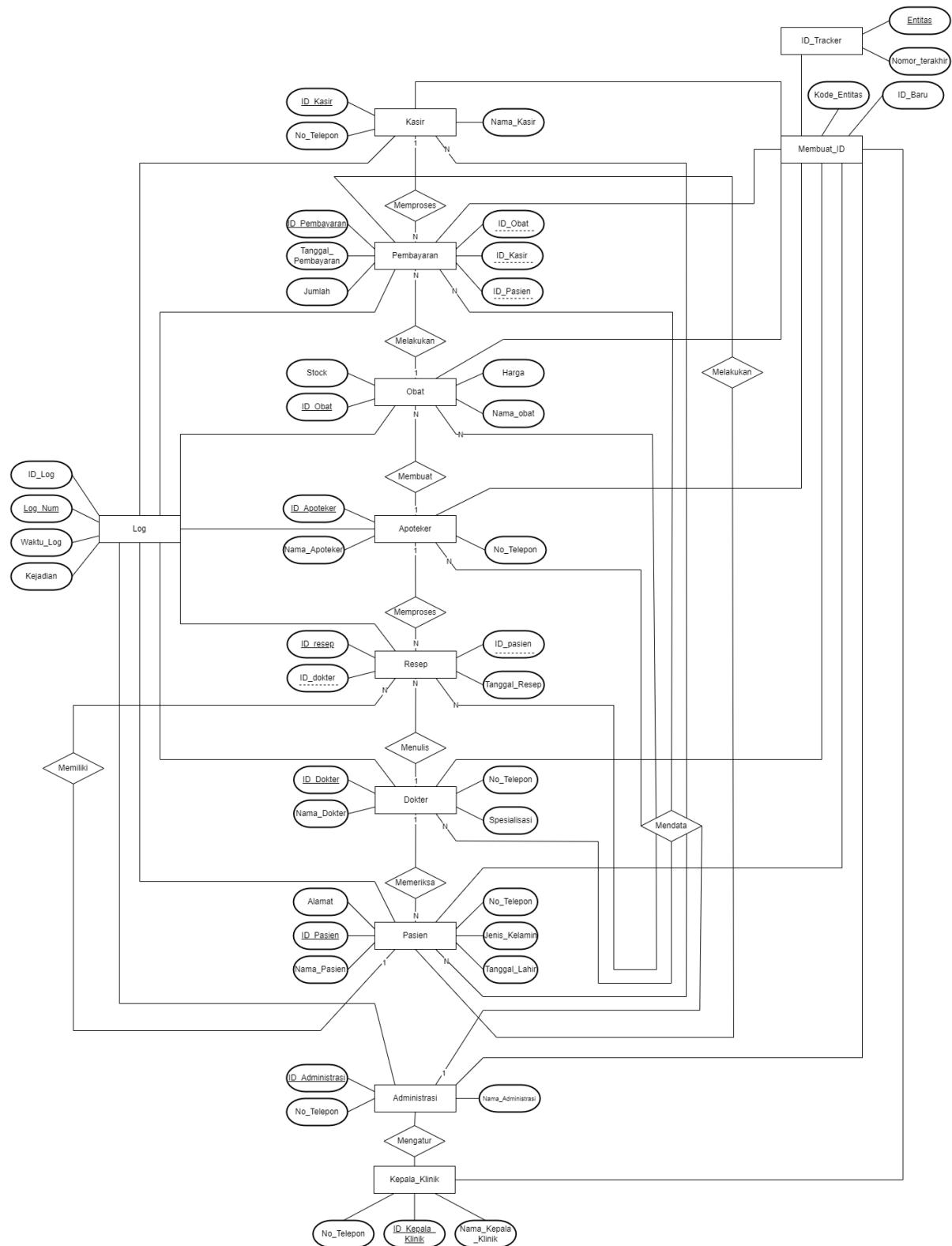
Tujuan dari tugas ini adalah untuk merancang dan mengembangkan sistem database yang mampu mengelola data klinik secara efisien dan terstruktur. Sistem ini akan diimplementasikan untuk mendukung berbagai aspek operasional klinik, termasuk pendaftaran pasien, pencatatan rekam medis, pengelolaan jadwal dokter, dan inventarisasi obat. Salah satu tujuan utama adalah meningkatkan efisiensi operasional dengan mengurangi ketergantungan pada pencatatan manual yang rentan terhadap kesalahan, serta memastikan keamanan data melalui mekanisme perlindungan yang kuat. Selain itu, sistem ini juga diharapkan dapat menyediakan akses informasi yang cepat dan akurat bagi staf klinik, sehingga pelayanan kepada pasien dapat ditingkatkan.

Manfaat dari tugas ini sangat beragam. Dengan adanya sistem database yang efisien, kualitas pelayanan di klinik dapat meningkat karena proses administrasi dan pencatatan menjadi lebih cepat dan tepat. Efisiensi waktu dan biaya juga dapat tercapai dengan mengurangi kebutuhan untuk pencatatan manual yang memakan waktu dan sumber daya. Risiko kesalahan pencatatan akan berkurang secara signifikan, berkat sistem otomatis yang mengurangi keterlibatan manusia dalam proses pengelolaan data. Integrasi data antar bagian dalam klinik, seperti administrasi, medis, dan apotek, akan lebih baik, sehingga koordinasi dan komunikasi internal menjadi lebih lancar. Keamanan dan kerahasiaan data pasien juga akan terjaga, melindungi informasi sensitif dari akses yang tidak sah. Kemudahan akses informasi bagi staf klinik memungkinkan mereka untuk mendapatkan data penting dengan cepat, yang sangat bermanfaat dalam situasi darurat medis. Akhirnya, data yang terorganisir dan akurat akan mendukung manajemen klinik dalam pengambilan keputusan yang lebih baik terkait operasional dan strategi pelayanan kesehatan.

BAB II

PERANCANGAN SISTEM

A. Entity Relational Diagram (ERD)



Berikut ini penjelasan mengenai entitas beserta atribut dari ERD di atas.

1. Dokter:
 - a. ID_Dokter: ID untuk dokter sebagai primary key,
 - b. Nama_Dokter: Nama lengkap dokter,
 - c. Spesialisasi: Bidang spesialisasi/ahli dari dokter, dan
 - d. No_Telepon_Dokter: Informasi nomor telepon untuk kontak dokter.
2. Administrasi:
 - a. ID_Administrasi: ID untuk dokter sebagai primary key,
 - b. Nama_Administrasi: Nama lengkap administrasi,dan
 - c. No_Telepon_Administrasi: Informasi nomor telepon untuk kontak administrasi.
3. Pasien:
 - a. ID_Pasien: ID untuk pasien sebagai primary key,
 - b. Nama_Pasien: Nama lengkap pasien,
 - c. Tanggal_Lahir: Tanggal lahir pasien,
 - d. Jenis_Kelamin: Jenis kelamin pasien,
 - e. No_Telepon_Pasien: Informasi nomor telepon untuk kontak pasien, dan
 - f. Alamat: Alamat tempat tinggal pasien.
4. Apoteker:
 - a. ID_Apoteker: ID untuk apoteker sebagai primary key,
 - b. Nama_Apoteker: Nama lengkap apoteker, dan
 - c. No_Telepon_Apoteker: Informasi nomor telepon untuk kontak apoteker.
5. Kasir:
 - a. ID_Kasir: ID untuk kasir sebagai primary key,
 - b. Nama_Kasir: Nama lengkap kasir, dan
 - c. No_Telepon_Kasir: Informasi nomor telepon untuk kontak kasir.
6. Kepala_Klinik:
 - a. ID_Kepala_Klinik: ID untuk kepala klinik sebagai primary key,
 - b. Nama_Kepala_Klinik: Nama lengkap kepala klinik, dan
 - c. No_Telepon_Kepala_Klinik: Informasi nomor telepon untuk kontak kepala klinik.

7. Obat:

- a. ID_Obat: ID untuk obat sebagai primary key,
- b. Nama_Obat: Nama obat,
- c. Harga: Harga obat, dan
- d. Stok: Banyaknya stok obat yang tersedia.

8. Resep:

- a. ID_Resep: ID untuk resep sebagai primary key,
- b. ID_Pasien: ID untuk pasien sebagai foreign key dari entitas Pasien,
- c. ID_Dokter: ID untuk dokter sebagai foreign key dari entitas Dokter, dan
- d. Tanggal_Resep: Tanggal dokter mengeluarkan resep untuk obat.

9. Resep Obat:

- a. ID_Resep: ID untuk resep sebagai primary key dan juga foreign key dari entitas Resep, dan
- b. ID_Obat: ID untuk obat sebagai primary key dan juga foreign key dari entitas Obat.

10. Pembayaran:

- a. ID_Pembayaran: ID untuk pembayaran sebagai primary key,
- b. ID_Pasien: ID untuk pasien sebagai foreign key dari entitas Pasien,
- c. ID_Kasir: ID untuk kasir sebagai foreign key dari entitas Kasir,
- d. Tanggal_Pembayaran: Tanggal dilakukannya pembayaran, dan
- e. Jumlah: Jumlah nominal yang dibayarkan.

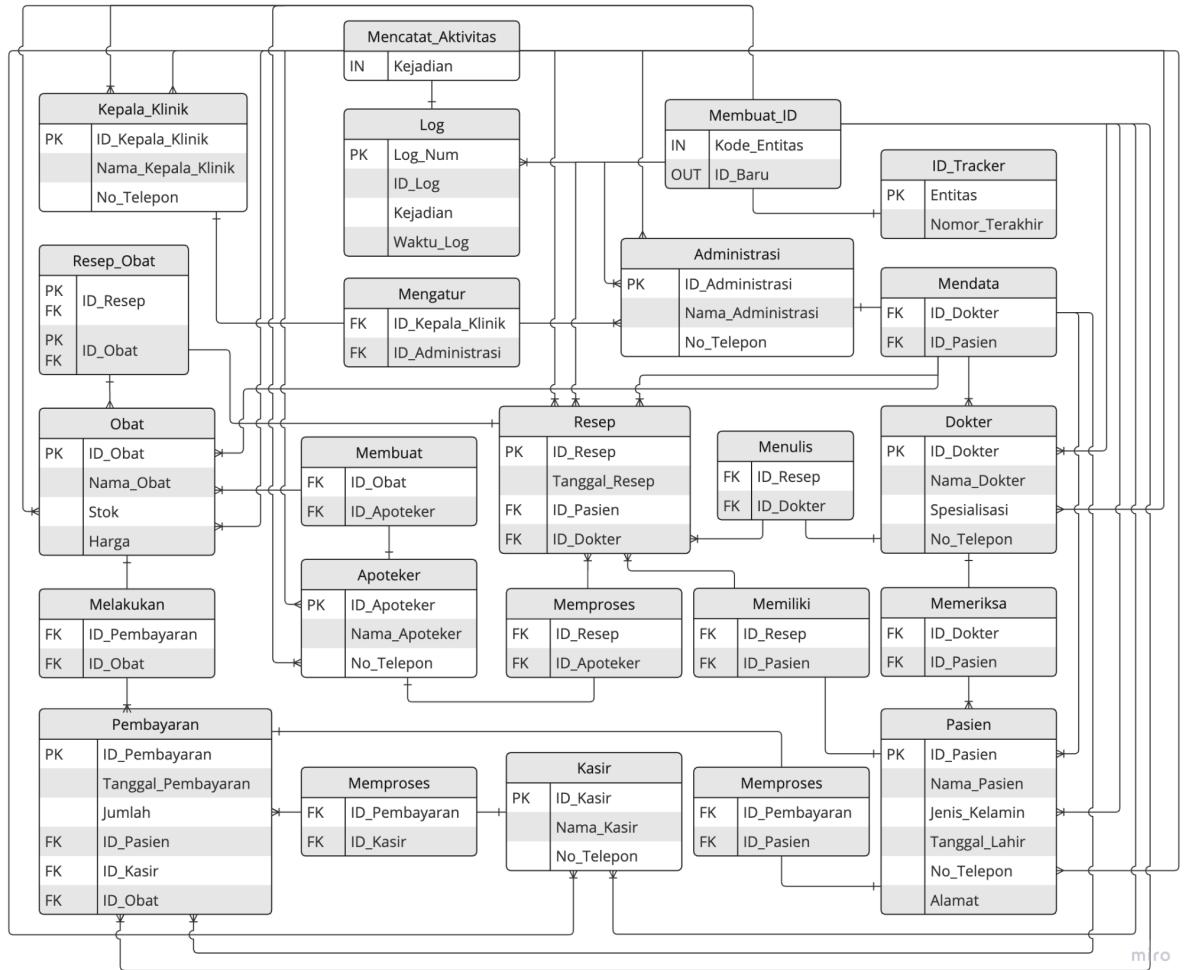
11. Log:

- a. Log_Num: Nomor urut otomatis untuk log sebagai primary key,
- b. ID_Log: ID untuk setiap log record,
- c. Kejadian: Deskripsi kejadian yang baru terjadi di setiap entitas, dan
- d. Waktu_Log: Tanggal dan waktu terjadinya suatu kejadian.

12. ID_Tracker:

- a. Entitas: Entitas yang baru saja dimanipulasi, sebagai primary key dan
- b. Nomor_Terakhir: Nomor terakhir ID masing-masing entitas.

B. Relational Model



Berikut ini penjelasan mengenai entitas beserta relasinya.

1. Dokter
 - a. menulis ‘Resep’ untuk pasien,
 - b. memeriksa ‘Pasien’ yang datang ke klinik, dan
 - c. didata oleh ‘Administrasi’.
2. Administrasi
 - a. diatur oleh ‘Kepala Klinik’ dan
 - b. mendata ‘Dokter’, ‘Pasien’, ‘Resep’, ‘Obat’, dan ‘Pembayaran’.
3. Pasien
 - a. melakukan ‘pembayaran’ untuk obat dan jasa medis.
 - b. menerima ‘Resep’ dari dokter setelah pemeriksaan.
4. Apoteker
 - a. membuat ‘Stock’ obat, dengan atribut stock dan harga.
 - b. memproses ‘Resep’ yang diberikan oleh dokter kepada pasien.

5. Kasir
 - a. memproses ‘pembayaran’ yang dilakukan oleh pasien, dengan atribut Tanggal_Pembayaran, Jumlah, ID_Pasien, dan ID_Obat.
6. Kepala_Klinik
 - a. memiliki otoritas untuk mengelola aktivitas klinik termasuk pengawasan terhadap administrasi.
7. Obat
 - a. dibuat oleh Apoteker dan terkait dengan Pembayaran
 - b. Melakukan pembayaran Obat di Pembayaran
8. Resep
 - c. ditulis oleh Dokter dan milik Pasien
 - d. dipakai oleh Apoteker untuk membuat Obat
 - e. terdiri dari satu atau lebih Obat
9. Resep_Obat
 - a. Membuat Resep dibuat untuk satu atau lebih obat.
 - b. Membuat obat ke Apoteker di Apoteker
 - c. Menulis Resep obat di Dokter
 - d. Memproses Resep obat di Apoteker
10. Pembayaran
 - a. diproses oleh Kasir dan dilakukan oleh Apoteker
11. Log
 - a. mencatat setiap aktivitas yang terjadi pada setiap entitas.
12. ID_Tracker
 - a. menyimpan kode entitas untuk dibuatkan ID-nya.

BAB III

IMPLEMENTASI DASAR

A. Data Definition Language (DDL)

1. Membuat database ‘Klinik’.

```
CREATE DATABASE Klinik;
```

```
USE Klinik;
```

```
5  CREATE DATABASE Klinik;
6  USE Klinik;
```

2. Membuat tabel ‘Dokter’.

```
CREATE TABLE Dokter (
```

```
    ID_Dokter VARCHAR(5) PRIMARY KEY NOT NULL,  
    Nama_Dokter VARCHAR(50) NOT NULL,  
    Spesialisasi VARCHAR(50),  
    No_Telepon_Dokter VARCHAR(15)
```

```
);
```

```
8  CREATE TABLE Dokter (
9      ID_Dokter VARCHAR(5) PRIMARY KEY,
10     Nama_Dokter VARCHAR(50) NOT NULL,
11     Spesialisasi VARCHAR(50),
12     No_Telepon_Dokter VARCHAR(15)
13 );
```

3. Membuat tabel ‘Administrasi’.

```
CREATE TABLE Administrasi (
```

```
    ID_Administrasi VARCHAR(5) PRIMARY KEY NOT NULL,  
    Nama_Administrasi VARCHAR(50) NOT NULL,  
    No_Telepon_Administrasi VARCHAR(15)
```

```
);
```

```
15  CREATE TABLE Administrasi (
16  |   ID_Administrasi VARCHAR(5) PRIMARY KEY,
17  |   Nama_Administrasi VARCHAR(50) NOT NULL,
18  |   No_Telepon_Administrasi VARCHAR(15)
19  );
```

4. Membuat tabel ‘Pasien’.

```
CREATE TABLE Pasien (
    ID_Pasien VARCHAR(5) PRIMARY KEY NOT NULL,
    Nama_Pasien VARCHAR(50) NOT NULL,
    Tanggal_Lahir DATE,
    Jenis_Kelamin VARCHAR(10),
    No_Telepon_Pasien VARCHAR(15),
    Alamat VARCHAR(100)
);
```

```
21  CREATE TABLE Pasien (
22  |   ID_Pasien VARCHAR(5) PRIMARY KEY,
23  |   Nama_Pasien VARCHAR(50) NOT NULL,
24  |   Tanggal_Lahir DATE,
25  |   Jenis_Kelamin VARCHAR(10),
26  |   No_Telepon_Pasien VARCHAR(15),
27  |   Alamat VARCHAR(100)
28  );
```

5. Membuat tabel ‘Apoteker’.

```
CREATE TABLE Apoteker (
    ID_Apoteker VARCHAR(5) PRIMARY KEY NOT NULL,
    Nama_Apoteker VARCHAR(50) NOT NULL,
    No_Telepon_Apoteker VARCHAR(15)
);
```

```
30  CREATE TABLE Apoteker (
31  |   ID_Apoteker VARCHAR(5) PRIMARY KEY,
32  |   Nama_Apoteker VARCHAR(50) NOT NULL,
33  |   No_Telepon_Apoteker VARCHAR(15)
34  );
```

6. Membuat tabel ‘Kasir’.

```
CREATE TABLE Kasir (
    ID_Kasir VARCHAR(5) PRIMARY KEY NOT NULL,
    Nama_Kasir VARCHAR(50) NOT NULL,
    No_Telepon_Kasir VARCHAR(15)
);
```

```
36  CREATE TABLE Kasir (
37      ID_Kasir VARCHAR(5) PRIMARY KEY,
38      Nama_Kasir VARCHAR(50) NOT NULL,
39      No_Telepon_Kasir VARCHAR(15)
40  );
```

7. Membuat tabel ‘Kepala_Klinik’.

```
CREATE TABLE Kepala_Klinik (
    ID_Kepala_Klinik VARCHAR(5) PRIMARY KEY NOT NULL,
    Nama_Kepala_Klinik VARCHAR(50) NOT NULL,
    No_Telepon_Kepala_Klinik VARCHAR(15)
);
```

```
42  CREATE TABLE Kepala_Klinik (
43      ID_Kepala_Klinik VARCHAR(5) PRIMARY KEY,
44      Nama_Kepala_Klinik VARCHAR(50) NOT NULL,
45      No_Telepon_Kepala_Klinik VARCHAR(15)
46  );
```

8. Membuat tabel ‘Obat’.

```
CREATE TABLE Obat (
    ID_Obat VARCHAR(5) PRIMARY KEY NOT NULL,
    Nama_Obat VARCHAR(50),
    Harga DECIMAL(10, 2),
    Stok INT
);
```

```
48  CREATE TABLE Obat (
49      ID_Obat VARCHAR(5) PRIMARY KEY,
50      Nama_Obat VARCHAR(50),
51      Harga DECIMAL(10, 2),
52      Stok INT
53  );
```

9. Membuat tabel ‘Resep’.

```
CREATE TABLE Resep (
    ID_Resep VARCHAR(5) PRIMARY KEY NOT NULL,
    ID_Pasien VARCHAR(5),
    ID_Dokter VARCHAR(5),
    Tanggal_Resep DATE,
    FOREIGN KEY (ID_Pasien) REFERENCES Pasien(ID_Pasien),
    FOREIGN KEY (ID_Dokter) REFERENCES Dokter(ID_Dokter)
);
```

```
55  CREATE TABLE Resep (
56  |   ID_Resep VARCHAR(5) PRIMARY KEY,
57  |   ID_Pasien VARCHAR(5),
58  |   ID_Dokter VARCHAR(5),
59  |   Tanggal_Resep DATE,
60  |   FOREIGN KEY (ID_Pasien) REFERENCES Pasien(ID_Pasien),
61  |   FOREIGN KEY (ID_Dokter) REFERENCES Dokter(ID_Dokter)
62  );
```

10. Membuat tabel ‘Resep_Obat’.

```
CREATE TABLE Resep_Obat (
    ID_Resep VARCHAR(5),
    ID_Obat VARCHAR(5),
    PRIMARY KEY (ID_Resep, ID_Obat),
    FOREIGN KEY (ID_Resep) REFERENCES Resep(ID_Resep),
    FOREIGN KEY (ID_Obat) REFERENCES Obat(ID_Obat)
);
```

```
64  CREATE TABLE Resep_Obat (
65  |   ID_Resep VARCHAR(5),
66  |   ID_Obat VARCHAR(5),
67  |   PRIMARY KEY (ID_Resep, ID_Obat),
68  |   FOREIGN KEY (ID_Resep) REFERENCES Resep(ID_Resep),
69  |   FOREIGN KEY (ID_Obat) REFERENCES Obat(ID_Obat)
70  );
```

11. Membuat tabel ‘Pembayaran’.

```
CREATE TABLE Pembayaran (
    ID_Pembayaran VARCHAR(5) PRIMARY KEY NOT NULL,
    ID_Pasien VARCHAR(5),
```

```

        ID_Kasir VARCHAR(5),
        Tanggal_Pembayaran DATE,
        Jumlah DECIMAL(10, 2),
        FOREIGN KEY (ID_Pasien) REFERENCES Pasien(ID_Pasien),
        FOREIGN KEY (ID_Kasir) REFERENCES Kasir(ID_Kasir)
    );

```

```

72  CREATE TABLE Pembayaran (
73  |   ID_Pembayaran VARCHAR(5) PRIMARY KEY,
74  |   ID_Pasien VARCHAR(5),
75  |   ID_Kasir VARCHAR(5),
76  |   Tanggal_Pembayaran DATE,
77  |   Jumlah DECIMAL(10, 2),
78  |   FOREIGN KEY (ID_Pasien) REFERENCES Pasien(ID_Pasien),
79  |   FOREIGN KEY (ID_Kasir) REFERENCES Kasir(ID_Kasir)
80  );

```

12. Membuat tabel ‘Log’.

```

CREATE TABLE Log (
    Log_Num INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
    ID_Log VARCHAR(8),
    Kejadian VARCHAR(255),
    Waktu_Log TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

```

```

82  CREATE TABLE Log (
83  |   Log_Num INT AUTO_INCREMENT PRIMARY KEY,
84  |   ID_Log VARCHAR(8),
85  |   Kejadian VARCHAR(255),
86  |   Waktu_Log TIMESTAMP DEFAULT CURRENT_TIMESTAMP
87  );

```

13. Membuat tabel ‘ID_Tracker’.

```

CREATE TABLE Pembayaran (
    Entitas VARCHAR(2) PRIMARY KEY NOT NULL,
    Nomor_Terakhir INT NOT NULL
);

```

```

89  CREATE TABLE ID_Tracker (
90  |   Entitas VARCHAR(2) PRIMARY KEY,
91  |   Nomor_Terakhir INT NOT NULL
92  );

```

B. Data Manipulation Language (DML)

1. Memasukkan record ke tabel ‘ID_Tracker’.

```
INSERT INTO ID_Tracker (Entitas, Nomor_Terakhir)
VALUES ('DR', 0), ('AD', 0), ('PS', 0), ('AP', 0), ('KS', 0),
       ('KK', 0), ('RE', 0), ('OB', 0), ('PM', 0), ('LG', 0);
```

```
473  INSERT INTO ID_Tracker (Entitas, Nomor_Terakhir)
474  VALUES  ('DR', 0), ('AD', 0), ('PS', 0), ('AP', 0), ('KS', 0),
475    |     ('KK', 0), ('RE', 0), ('OB', 0), ('PM', 0), ('LG', 0);
```

2. Memasukkan record ke tabel ‘Dokter’.

```
INSERT INTO Dokter (Nama_Dokter, Spesialisasi, No_Telepon_Dokter)
VALUES
      ('Dr. Asep', 'Kardiologi', '081234567890'),
      ('Dr. Budi', 'Pediatri', '081234567891'),
      ('Dr. Chandra', 'Bedah', '081234567892'),
      ('Dr. Darwin', 'Kulit', '081234567893'),
      ('Dr. Erni', 'THT', '081234567894'),
      ('Dr. Fatih', 'Mata', '081234567895'),
      ('Dr. Gita', 'Saraf', '081234567896'),
      ('Dr. Handani', 'Gigi', '081234567897'),
      ('Dr. Intan', 'Umum', '081234567898'),
      ('Dr. Joko', 'Psikiatri', '081234567899');
```

```
477  INSERT INTO Dokter (Nama_Dokter, Spesialisasi, No_Telepon_Dokter)
478  VALUES
479    ('Dr. Asep', 'Kardiologi', '081234567890'),
480    ('Dr. Budi', 'Pediatri', '081234567891'),
481    ('Dr. Chandra', 'Bedah', '081234567892'),
482    ('Dr. Darwin', 'Kulit', '081234567893'),
483    ('Dr. Erni', 'THT', '081234567894'),
484    ('Dr. Fatih', 'Mata', '081234567895'),
485    ('Dr. Gita', 'Saraf', '081234567896'),
486    ('Dr. Handani', 'Gigi', '081234567897'),
487    ('Dr. Intan', 'Umum', '081234567898'),
488    ('Dr. Joko', 'Psikiatri', '081234567899');
```

3. Memasukkan record ke tabel ‘Administrasi’.

```
INSERT INTO Administrasi (Nama_Administrasi, No_Telepon_Administrasi)
VALUES
```

```

('Andi', '081234567800'),
('Burhan', '081234567801'),
('Citra', '081234567802'),
('Dewi', '081234567803'),
('Eka', '081234567804'),
('Fajar', '081234567805'),
('Gina', '081234567806'),
('Heri', '081234567807'),
('Indah', '081234567808'),
('Joni', '081234567809');

```

```

490  INSERT INTO Administrasi (Nama_Administrasi, No_Telepon_Administrasi)
491  VALUES
492  ('Andi', '081234567800'),
493  ('Burhan', '081234567801'),
494  ('Citra', '081234567802'),
495  ('Dewi', '081234567803'),
496  ('Eka', '081234567804'),
497  ('Fajar', '081234567805'),
498  ('Gina', '081234567806'),
499  ('Heri', '081234567807'),
500  ('Indah', '081234567808'),
501  ('Joni', '081234567809');

```

4. Memasukkan record ke tabel ‘Pasien’.

```

INSERT INTO Pasien
(Nama_Pasien, Tanggal_Lahir, Jenis_Kelamin, No_Telepon_Pasien, Alamat)
VALUES
('Ali', '1990-01-01', 'Laki-laki', '081234567810', 'Jl. Cimindi'),
('Berry', '1985-02-02', 'Laki-laki', '081234567811', 'Jl. Nasional 3'),
('Cinta', '1995-03-03', 'Perempuan', '081234567812', 'Jl. semoga'),
('Diwanto', '2000-04-04', 'Perempuan', '081234567813', 'Jl. sukabrius'),
('Emma', '1975-05-05', 'Laki-laki', '081234567814', 'Jl. BKT'),
('Fajar', '1992-06-06', 'Laki-laki', '081234567815', 'Jl. cibiru'),
('Gery', '1998-07-07', 'Perempuan', '081234567816', 'Jl. jalan'),
('Hilmi', '1980-08-08', 'Laki-laki', '081234567817', 'Jl. Moh. Toha'),
('Ira', '1994-09-09', 'Perempuan', '081234567818', 'Jl. Rahmat'),
('Juandi', '1991-10-10', 'Laki-laki', '081234567819', 'Jl. sukapura');

```

```

503  INSERT INTO Pasien
504    (Nama_Pasien, Tanggal_Lahir, Jenis_Kelamin, No_Telepon_Pasien, Alamat)
505    VALUES
506    ('Ali', '1990-01-01', 'Laki-laki', '081234567810', 'Jl. Cimindi'),
507    ('Berry', '1985-02-02', 'Laki-laki', '081234567811', 'Jl. Nasional 3'),
508    ('Cinta', '1995-03-03', 'Perempuan', '081234567812', 'Jl. semoga'),
509    ('Diwanto', '2000-04-04', 'Perempuan', '081234567813', 'Jl. sukabrius'),
510    ('Emma', '1975-05-05', 'Laki-laki', '081234567814', 'Jl. BKT'),
511    ('Fajar', '1992-06-06', 'Laki-laki', '081234567815', 'Jl. cibiru'),
512    ('Gery', '1998-07-07', 'Perempuan', '081234567816', 'Jl. jalan'),
513    ('Hilmi', '1980-08-08', 'Laki-laki', '081234567817', 'Jl. Moh. Toha'),
514    ('Ira', '1994-09-09', 'Perempuan', '081234567818', 'Jl. Rahmat'),
515    ('Juandi', '1991-10-10', 'Laki-laki', '081234567819', 'Jl. sukapura');

```

5. Memasukkan record ke tabel ‘Apoteker’.

```

INSERT INTO Apoteker (Nama_Apoteker, No_Telepon_Apoteker)
VALUES
('Amir', '081234567820'),
('Buston', '081234567821'),
('Carista', '081234567822'),
('Darti', '081234567823'),
('Ena', '081234567824'),
('Fajar', '081234567825'),
('Gitawa', '081234567826'),
('Hafid', '081234567827'),
('Ilham', '081234567828'),
('Jarfis', '081234567829');

```

```

517  INSERT INTO Apoteker (Nama_Apoteker, No_Telepon_Apoteker)
518  VALUES
519  ('Amir', '081234567820'),
520  ('Buston', '081234567821'),
521  ('Carista', '081234567822'),
522  ('Darti', '081234567823'),
523  ('Ena', '081234567824'),
524  ('Fajar', '081234567825'),
525  ('Gitawa', '081234567826'),
526  ('Hafid', '081234567827'),
527  ('Ilham', '081234567828'),
528  ('Jarfis', '081234567829');

```

6. Memasukkan record ke tabel ‘Kasir’.

```

INSERT INTO Kasir (Nama_Kasir, No_Telepon_Kasir)
VALUES

```

('Ahmad', '081234567830'),
('Basuki', '081234567831'),
('Calvin', '081234567832'),
('Desti', '081234567833'),
('Edo', '081234567834'),
('Farhan', '081234567835'),
('Genos', '081234567836'),
('Hesti', '081234567837'),
('Iki', '081234567838'),
('Jarip', '081234567839');

```
530  INSERT INTO Kasir (Nama_Kasir, No_Telepon_Kasir)  
531  VALUES  
532  ('Ahmad', '081234567830'),  
533  ('Basuki', '081234567831'),  
534  ('Calvin', '081234567832'),  
535  ('Desti', '081234567833'),  
536  ('Edo', '081234567834'),  
537  ('Farhan', '081234567835'),  
538  ('Genos', '081234567836'),  
539  ('Hesti', '081234567837'),  
540  ('Iki', '081234567838'),  
541  ('Jarip', '081234567839');
```

7. Memasukkan record ke tabel ‘Kepala_Klinik’.

```
INSERT INTO Kepala_Klinik  
(Nama_Kepala_Klinik, No_Telepon_Kepala_Klinik)  
VALUES  
('Arif', '081234567840'),  
('Ben', '081234567841'),  
('Cici', '081234567842'),  
('Delima', '081234567843'),  
('Eko', '081234567844'),  
('Faisal', '081234567845'),  
('Gianis', '081234567846'),  
('Heru', '081234567847'),  
('Ivan', '081234567848'),  
('Justin', '081234567849');
```

```
543  INSERT INTO Kepala_Klinik (Nama_Kepala_Klinik, No_Telepon_Kepala_Klinik)
544  VALUES
545  ('Arif', '081234567840'),
546  ('Ben', '081234567841'),
547  ('Cici', '081234567842'),
548  ('Delima', '081234567843'),
549  ('Eko', '081234567844'),
550  ('Faisal', '081234567845'),
551  ('Gianis', '081234567846'),
552  ('Heru', '081234567847'),
553  ('Ivan', '081234567848'),
554  ('Justin', '081234567849');
```

8. Memasukkan record ke tabel ‘Obat’.

```
INSERT INTO Obat (Nama_Obat, Harga, Stok)
```

```
VALUES
```

```
('Paracetamol', 5000, 100),
('Amoxicillin', 10000, 200),
('Ibuprofen', 7500, 150),
('Vitamin C', 3000, 250),
('Cetirizine', 5000, 100),
('Antasida', 2000, 300),
('Loratadine', 6000, 180),
('Omeprazole', 8000, 120),
('Metformin', 10000, 100),
('Amlodipine', 9000, 140);
```

```
556  INSERT INTO Obat (Nama_Obat, Harga, Stok)
557  VALUES
558  ('Paracetamol', 5000, 100),
559  ('Amoxicillin', 10000, 200),
560  ('Ibuprofen', 7500, 150),
561  ('Vitamin C', 3000, 250),
562  ('Cetirizine', 5000, 100),
563  ('Antasida', 2000, 300),
564  ('Loratadine', 6000, 180),
565  ('Omeprazole', 8000, 120),
566  ('Metformin', 10000, 100),
567  ('Amlodipine', 9000, 140);
```

9. Memasukkan record ke tabel ‘Resep’.

```
INSERT INTO Resep (ID_Pasien, ID_Dokter, Tanggal_Resep)
```

```
VALUES
```

```
('PS001', 'DR001', '2023-02-03'),
```

```
('PS002', 'DR002', '2023-03-28'),  
('PS003', 'DR003', '2023-04-10'),  
('PS004', 'DR004', '2023-05-30'),  
('PS005', 'DR005', '2023-06-06'),  
('PS006', 'DR006', '2023-06-08'),  
('PS007', 'DR007', '2023-07-19'),  
('PS008', 'DR008', '2023-08-25'),  
('PS009', 'DR009', '2022-10-23'),  
('PS010', 'DR010', '2022-12-02');
```

```
569  INSERT INTO Resep (ID_Pasien, ID_Dokter, Tanggal_Resep)  
570  VALUES  
571  ('PS001', 'DR001', '2023-02-03'),  
572  ('PS002', 'DR002', '2023-03-28'),  
573  ('PS003', 'DR003', '2023-04-10'),  
574  ('PS004', 'DR004', '2023-05-30'),  
575  ('PS005', 'DR005', '2023-06-06'),  
576  ('PS006', 'DR006', '2023-06-08'),  
577  ('PS007', 'DR007', '2023-07-19'),  
578  ('PS008', 'DR008', '2023-08-25'),  
579  ('PS009', 'DR009', '2022-10-23'),  
580  ('PS010', 'DR010', '2022-12-02');
```

10. Memasukkan record ke tabel ‘Resep_Obat’.

```
INSERT INTO Resep_Obat (ID_Resep, ID_Obat)  
VALUES  
('RE001', 'OB001'), ('RE001', 'OB002'),  
('RE002', 'OB003'), ('RE002', 'OB004'),  
('RE003', 'OB005'), ('RE003', 'OB006'),  
('RE004', 'OB007'), ('RE004', 'OB008'),  
('RE004', 'OB009'), ('RE005', 'OB010'),  
('RE006', 'OB001'), ('RE006', 'OB002'),  
('RE007', 'OB003'), ('RE007', 'OB004'),  
('RE008', 'OB005'), ('RE008', 'OB006'),  
('RE009', 'OB007'), ('RE009', 'OB008'),  
('RE010', 'OB009'), ('RE010', 'OB010');
```

```

582  INSERT INTO Resep_Obat (ID_Resep, ID_Obat)
583  VALUES
584  ('RE001', 'OB001'), ('RE001', 'OB002'),
585  ('RE002', 'OB003'), ('RE002', 'OB004'),
586  ('RE003', 'OB005'), ('RE003', 'OB006'),
587  ('RE004', 'OB007'), ('RE004', 'OB008'),
588  ('RE005', 'OB009'), ('RE005', 'OB010'),
589  ('RE006', 'OB001'), ('RE006', 'OB002'),
590  ('RE007', 'OB003'), ('RE007', 'OB004'),
591  ('RE008', 'OB005'), ('RE008', 'OB006'),
592  ('RE009', 'OB007'), ('RE009', 'OB008'),
593  ('RE010', 'OB009'), ('RE010', 'OB010');

```

11. Memasukkan record ke tabel ‘Pembayaran’.

```

INSERT INTO Pembayaran
(ID_Pasien, ID_Kasir, Tanggal_Pembayaran, Jumlah)
VALUES
('PS001', 'KS001', '2023-02-03', 50000),
('PS002', 'KS002', '2023-03-28', 75000),
('PS003', 'KS003', '2023-04-10', 60000),
('PS004', 'KS004', '2023-05-30', 85000),
('PS005', 'KS005', '2023-06-06', 90000),
('PS006', 'KS006', '2023-06-08', 55000),
('PS007', 'KS007', '2023-07-19', 70000),
('PS008', 'KS008', '2023-08-25', 65000),
('PS009', 'KS009', '2022-10-23', 80000),
('PS010', 'KS010', '2022-12-02', 95000);

```

```

595  INSERT INTO Pembayaran (ID_Pasien, ID_Kasir, Tanggal_Pembayaran, Jumlah)
596  VALUES
597  ('PS001', 'KS001', '2023-02-03', 50000),
598  ('PS002', 'KS002', '2023-03-28', 75000),
599  ('PS003', 'KS003', '2023-04-10', 60000),
600  ('PS004', 'KS004', '2023-05-30', 85000),
601  ('PS005', 'KS005', '2023-06-06', 90000),
602  ('PS006', 'KS006', '2023-06-08', 55000),
603  ('PS007', 'KS007', '2023-07-19', 70000),
604  ('PS008', 'KS008', '2023-08-25', 65000),
605  ('PS009', 'KS009', '2022-10-23', 80000),
606  ('PS010', 'KS010', '2022-12-02', 95000);

```

BAB IV

IMPLEMENTASI LANJUTAN

A. Stored Procedure

- 1. Membuat procedure ‘Generate_ID’ untuk membuat ID otomatis untuk setiap record di masing-masing tabel entitas.**

DELIMITER //

```
CREATE PROCEDURE Generate_ID(
    IN Kode_Entitas VARCHAR(2),
    OUT ID_Baru VARCHAR(5)
)
BEGIN
    DECLARE P_Nomor_Terakhir INT;
    DECLARE P_Nomor_Baru INT;

    SELECT Nomor_Terakhir INTO P_Nomor_Terakhir
    FROM ID_Tracker
    WHERE Entitas = Kode_Entitas
    FOR UPDATE;

    SET P_Nomor_Baru = P_Nomor_Terakhir + 1;

    UPDATE ID_Tracker
    SET Nomor_Terakhir = P_Nomor_Baru
    WHERE Entitas = Kode_Entitas;

    SET ID_Baru = CONCAT(Kode_Entitas, LPAD(P_Nomor_Baru, 3, '0'));

END //
```

DELIMITER ;

```

94  -- Stored Procedure untuk Generate Custom ID
95  DELIMITER //
96
97  CREATE PROCEDURE Generate_ID(
98      IN Kode_Entitas VARCHAR(2),
99      OUT ID_Baru VARCHAR(5)
100 )
101 BEGIN
102     DECLARE P_Nomor_Terakhir INT;
103     DECLARE P_Nomor_Baru INT;
104
105     SELECT Nomor_Terakhir INTO P_Nomor_Terakhir
106     FROM ID_Tracker
107     WHERE Entitas = Kode_Entitas
108     FOR UPDATE;
109
110     SET P_Nomor_Baru = P_Nomor_Terakhir + 1;
111
112     UPDATE ID_Tracker
113     SET Nomor_Terakhir = P_Nomor_Baru
114     WHERE Entitas = Kode_Entitas;
115
116     SET ID_Baru = CONCAT(Kode_Entitas, LPAD(P_Nomor_Baru, 3, '0'));
117 END //
118
119 DELIMITER ;

```

2. Membuat procedure ‘Tambah_Obat_Baru’ untuk menambahkan obat baru ke dalam tabel ‘Obat’.

DELIMITER //

CREATE PROCEDURE Tambah_Obat_Baru (

IN P_ID_Obat VARCHAR(5),
 IN P_Nama_Obat VARCHAR(50),
 IN P_Harga DECIMAL(10, 2),
 IN P_Stok INT

)

BEGIN

DECLARE Obat_ada INT;

SELECT COUNT(*) INTO Obat_ada
 FROM Obat
 WHERE ID_Obat = P_ID_Obat;

IF Obat_ada = 0 THEN

```

        INSERT INTO Obat (ID_Obat, Nama_Obat, Harga, Stok)
        VALUES (P_ID_Obat, P_Nama_Obat, P_Harga, P_Stok);

        SET @status = 'Obat baru berhasil ditambahkan.';

        ELSE
            SET @status = 'Obat dengan ID tersebut sudah ada.';

        END IF;

        SELECT @status;
    END //

DELIMITER ;

```

```

318 -- Stored Procedure terkait Obat
319 DELIMITER //
320
321 CREATE PROCEDURE Tambah_Obat_Baru (
322     IN P_ID_Obat VARCHAR(5),
323     IN P_Nama_Obat VARCHAR(50),
324     IN P_Harga DECIMAL(10, 2),
325     IN P_Stok INT
326 )
327 BEGIN
328     DECLARE Obat_ada INT;
329
330     SELECT COUNT(*) INTO Obat_ada
331     FROM Obat
332     WHERE ID_Obat = P_ID_Obat;
333
334     IF Obat_ada = 0 THEN
335         INSERT INTO Obat (ID_Obat, Nama_Obat, Harga, Stok)
336             VALUES (P_ID_Obat, P_Nama_Obat, P_Harga, P_Stok);
337         SET @status = 'Obat baru berhasil ditambahkan.';
338     ELSE
339         SET @status = 'Obat dengan ID tersebut sudah ada.';
340     END IF;
341
342     SELECT @status;
343 END //
344
345 DELIMITER ;

```

3. Membuat procedure ‘Tambah_Stok_Obat’ untuk menambahkan stok pada obat yang sudah terdaftar di tabel ‘Obat’.

DELIMITER //

CREATE PROCEDURE Tambah_Stok_Obat (

```

    IN P_ID_Obat VARCHAR(5),
    IN P_Stok INT
)
BEGIN
    DECLARE Obat_ada INT;

    SELECT COUNT(*) INTO Obat_ada
    FROM Obat
    WHERE ID_Obat = P_ID_Obat;

    IF Obat_ada > 0 THEN
        UPDATE Obat
        SET Stok = Stok + P_Stok
        WHERE ID_Obat = P_ID_Obat;

        SET @status = 'Stok obat berhasil ditambahkan。';
    ELSE
        SET @status = 'Obat tidak ditemukan。';
    END IF;

    SELECT @status;
END //

DELIMITER ;

```

```

347  DELIMITER //
348
349  CREATE PROCEDURE Tambah_Stok_Obat (
350      IN P_ID_Obat VARCHAR(5),
351      IN P_Stok INT
352  )
353  BEGIN
354      DECLARE Obat_ada INT;
355
356      SELECT COUNT(*) INTO Obat_ada
357      FROM Obat
358      WHERE ID_Obat = P_ID_Obat;
359
360      IF Obat_ada > 0 THEN
361          UPDATE Obat
362              SET Stok = Stok + P_Stok
363              WHERE ID_Obat = P_ID_Obat;
364          SET @status = 'Stok obat berhasil ditambahkan.';
365      ELSE
366          SET @status = 'Obat tidak ditemukan.';
367      END IF;
368
369      SELECT @status;
370  END //
371
372  DELIMITER ;

```

4. Membuat procedure ‘Kurangi_Stok_Obat’ untuk mengurangi stok pada obat yang sudah terdaftar di tabel ‘Obat’.

DELIMITER //

```

CREATE PROCEDURE Kurangi_Stok_Obat (
    IN P_ID_Obat VARCHAR(5),
    IN P_Stok INT
)
BEGIN
    DECLARE Obat_ada INT;
    DECLARE Stok_cukup INT;

    SELECT COUNT(*) INTO Obat_ada
    FROM Obat
    WHERE ID_Obat = P_ID_Obat;

    SELECT Stok INTO Stok_cukup
    FROM Obat
    WHERE ID_Obat = P_ID_Obat;

```

```
WHERE ID_Obat = P_ID_Obat;

IF Obat_ada > 0 THEN
    IF Stok_cukup >= P_Stok THEN
        UPDATE Obat
        SET Stok = Stok - P_Stok
        WHERE ID_Obat = P_ID_Obat;

        SET @status = 'Stok obat berhasil dikurangi。';
    ELSE
        SET @status = 'Stok obat tidak cukup。';
    END IF;
ELSE
    SET @status = 'Obat tidak ditemukan。';
END IF;

SELECT @status;
END //


DELIMITER ;
```

```

374  DELIMITER //
375
376  CREATE PROCEDURE Kurangi_Stok_Obat (
377      IN P_ID_Obat VARCHAR(5),
378      IN P_Stok INT
379  )
380  BEGIN
381      DECLARE Obat_ada INT;
382      DECLARE Stok_cukup INT;
383
384      SELECT COUNT(*) INTO Obat_ada
385      FROM Obat
386      WHERE ID_Obat = P_ID_Obat;
387
388      SELECT Stok INTO Stok_cukup
389      FROM Obat
390      WHERE ID_Obat = P_ID_Obat;
391
392      IF Obat_ada > 0 THEN
393          IF Stok_cukup >= P_Stok THEN
394              UPDATE Obat
395                  SET Stok = Stok - P_Stok
396                  WHERE ID_Obat = P_ID_Obat;
397              SET @status = 'Stok obat berhasil dikurangi.';
398          ELSE
399              SET @status = 'Stok obat tidak cukup.';
400          END IF;
401      ELSE
402          SET @status = 'Obat tidak ditemukan.';
403      END IF;
404
405      SELECT @status;
406  END //
407
408  DELIMITER ;

```

- 5. Membuat procedure ‘Lihat_RO_by_ID’ untuk melihat resep obat pasien berdasarkan ID_Pasien.**

DELIMITER //

```

CREATE PROCEDURE Lihat_RO_by_ID (
    IN P_ID_Pasien VARCHAR(5)
)
BEGIN
    DECLARE Pasien_ada INT;

    SELECT COUNT(*) INTO Pasien_ada
    FROM Pasien
    WHERE ID_Pasien = P_ID_Pasien;

```

```

IF Pasien_ada > 0 THEN
    SELECT R.ID_Resep, R.Tanggal_Resep, D.Nama_Dokter,
GROUP_CONCAT(O.Nama_Obat SEPARATOR ',') AS Daftar_Obat
    FROM Resep R
    JOIN Dokter D ON R.ID_Dokter = D.ID_Dokter
    JOIN Resep_Obat RO ON R.ID_Resep = RO.ID_Resep
    JOIN Obat O ON RO.ID_Obat = O.ID_Obat
    WHERE R.ID_Pasien = P_ID_Pasien
    GROUP BY R.ID_Resep;
ELSE
    SET @status = 'Pasien dengan ID tersebut tidak terdaftar。';
    SELECT @status;
END IF;
END //


DELIMITER ;

```

```

410  -- Stored Procedure untuk Lihat Resep Obat Pasien
411  DELIMITER //
412
413  CREATE PROCEDURE Lihat_RO_by_ID (
414      IN P_ID_Pasien VARCHAR(5)
415  )
416  BEGIN
417      DECLARE Pasien_ada INT;
418
419      SELECT COUNT(*) INTO Pasien_ada
420      FROM Pasien
421      WHERE ID_Pasien = P_ID_Pasien;
422
423      IF Pasien_ada > 0 THEN
424          SELECT R.ID_Resep, R.Tanggal_Resep, D.Nama_Dokter,
425              GROUP_CONCAT(O.Nama_Obat SEPARATOR ', ') AS Daftar_Obat
426          FROM Resep R
427          JOIN Dokter D ON R.ID_Dokter = D.ID_Dokter
428          JOIN Resep_Obat RO ON R.ID_Resep = RO.ID_Resep
429          JOIN Obat O ON RO.ID_Obat = O.ID_Obat
430          WHERE R.ID_Pasien = P_ID_Pasien
431          GROUP BY R.ID_Resep;
432      ELSE
433          SET @status = 'Pasien dengan ID tersebut tidak terdaftar.';
434          SELECT @status;
435      END IF;
436  END //
437
438  DELIMITER ;

```

6. Membuat procedure ‘Lihat_RO_by_Nama’ untuk melihat resep obat pasien berdasarkan Nama_Pasien.

DELIMITER //

```

CREATE PROCEDURE Lihat_RO_by_Nama (
    IN P_Nama_Pasien VARCHAR(50)
)
BEGIN
    DECLARE Pasien_ada INT;

    SELECT COUNT(*) INTO Pasien_ada
    FROM Pasien
    WHERE Nama_Pasien = P_Nama_Pasien;

    IF Pasien_ada > 0 THEN
        SELECT R.ID_Resep, R.Tanggal_Resep, D.Nama_Dokter

```

```

        FROM Resep R
        JOIN Dokter D ON R.ID_Dokter = D.ID_Dokter
        WHERE R.ID_Pasien =
            SELECT ID_Pasien
            FROM Pasien
            WHERE Nama_Pasien = P_Nama_Pasien
        );
    ELSE
        SET @status = 'Pasien dengan nama tersebut tidak terdaftar.';
        SELECT @status;
    END IF;
END //
```

DELIMITER ;

```

440  DELIMITER //
441
442  CREATE PROCEDURE Lihat_R0_by>Nama (
443      IN P_Nama_Pasien VARCHAR(50)
444  )
445  BEGIN
446      DECLARE Pasien_ada INT;
447
448      SELECT COUNT(*) INTO Pasien_ada
449      FROM Pasien
450      WHERE Nama_Pasien = P_Nama_Pasien;
451
452      IF Pasien_ada > 0 THEN
453          SELECT R.ID_Resep, R.Tanggal_Resep, D.Nama_Dokter
454          FROM Resep R
455          JOIN Dokter D ON R.ID_Dokter = D.ID_Dokter
456          WHERE R.ID_Pasien =
457              SELECT ID_Pasien
458              FROM Pasien
459              WHERE Nama_Pasien = P_Nama_Pasien
460      );
461      ELSE
462          SET @status = 'Pasien dengan nama tersebut tidak terdaftar.';
463          SELECT @status;
464      END IF;
465  END //
466
467  DELIMITER ;
```

B. Trigger

1. Membuat trigger ‘Before_Insert_DR’ untuk memanggil procedure ‘Generate_ID’ secara otomatis setiap setiap kali terjadi penambahan record di tabel ‘Dokter’.

DELIMITER //

```
CREATE TRIGGER Before_Insert_DR
BEFORE INSERT ON Dokter
FOR EACH ROW
BEGIN
    DECLARE ID_Baru VARCHAR(5);

    CALL Generate_ID('DR', ID_Baru);

    SET NEW.ID_Dokter = ID_Baru;
END //
```

DELIMITER ;

```
121  -- Trigger untuk Generate Custom ID
122  DELIMITER //
123
124  CREATE TRIGGER Before_Insert_DR
125  BEFORE INSERT ON Dokter
126  FOR EACH ROW
127  BEGIN
128      DECLARE ID_Baru VARCHAR(5);
129
130      CALL Generate_ID('DR', ID_Baru);
131
132      SET NEW.ID_Dokter = ID_Baru;
133  END //
134
135  DELIMITER ;
```

2. Membuat trigger ‘Before_Insert_AD’ untuk memanggil procedure ‘Generate_ID’ secara otomatis setiap setiap kali terjadi penambahan record di tabel ‘Administrasi’.

DELIMITER //

```

CREATE TRIGGER Before_Insert_AD
BEFORE INSERT ON Administrasi
FOR EACH ROW
BEGIN
    DECLARE ID_Baru VARCHAR(5);

    CALL Generate_ID('AD', ID_Baru);

    SET NEW.ID_Administrasi = ID_Baru;
END //

DELIMITER ;

```

```

137  DELIMITER //
138
139  CREATE TRIGGER Before_Insert_AD
140  BEFORE INSERT ON Administrasi
141  FOR EACH ROW
142  BEGIN
143      DECLARE ID_Baru VARCHAR(5);
144
145      CALL Generate_ID('AD', ID_Baru);
146
147      SET NEW.ID_Administrasi = ID_Baru;
148  END //
149
150  DELIMITER ;

```

3. Membuat trigger ‘Before_Insert_PS’ untuk memanggil procedure ‘Generate_ID’ secara otomatis setiap setiap kali terjadi penambahan record di tabel ‘Pasien’.

DELIMITER //

```

CREATE TRIGGER Before_Insert_PS
BEFORE INSERT ON Pasien
FOR EACH ROW
BEGIN
    DECLARE ID_Baru VARCHAR(5);

```

```
CALL Generate_ID('PS', ID_Baru);
```

```
SET NEW.ID_Pasien = ID_Baru;  
END //
```

```
DELIMITER ;
```

```
152  DELIMITER //  
153  
154  CREATE TRIGGER Before_Insert_PS  
155  BEFORE INSERT ON Pasien  
156  FOR EACH ROW  
157  BEGIN  
158      DECLARE ID_Baru VARCHAR(5);  
159  
160      CALL Generate_ID('PS', ID_Baru);  
161  
162      SET NEW.ID_Pasien = ID_Baru;  
163  END //  
164  
165  DELIMITER ;
```

4. Membuat trigger ‘Before_Insert_AP’ untuk memanggil procedure ‘Generate_ID’ secara otomatis setiap setiap kali terjadi penambahan record di tabel ‘Apoteker’.

```
DELIMITER //
```

```
CREATE TRIGGER Before_Insert_AP
```

```
BEFORE INSERT ON Apoteker
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    DECLARE ID_Baru VARCHAR(5);
```

```
    CALL Generate_ID('AP', ID_Baru);
```

```
    SET NEW.ID_Apoteker = ID_Baru;
```

```
END //
```

```
DELIMITER ;
```

```
167  DELIMITER //
168
169  CREATE TRIGGER Before_Insert_AP
170  BEFORE INSERT ON Apoteker
171  FOR EACH ROW
172  BEGIN
173      DECLARE ID_Baru VARCHAR(5);
174
175      CALL Generate_ID('AP', ID_Baru);
176
177      SET NEW.ID_Apoteker = ID_Baru;
178  END //
179
180  DELIMITER ;
```

5. Membuat trigger ‘Before_Insert_KK’ untuk memanggil procedure ‘Generate_ID’ secara otomatis setiap setiap kali terjadi penambahan record di tabel ‘Kepala_Klinik’.

DELIMITER //

```
CREATE TRIGGER Before_Insert_KK
BEFORE INSERT ON Kepala_Klinik
FOR EACH ROW
BEGIN
    DECLARE ID_Baru VARCHAR(5);

    CALL Generate_ID('KK', ID_Baru);

    SET NEW.ID_Kepala_Klinik = ID_Baru;
END //

DELIMITER ;
```

```

182  DELIMITER //
183
184  CREATE TRIGGER Before_Insert_KK
185  BEFORE INSERT ON Kepala_Klinik
186  FOR EACH ROW
187  BEGIN
188      DECLARE ID_Baru VARCHAR(5);
189
190      CALL Generate_ID('KK', ID_Baru);
191
192      SET NEW.ID_Kepala_Klinik = ID_Baru;
193  END //
194
195  DELIMITER ;

```

6. Membuat trigger ‘Before_Insert_KS’ untuk memanggil procedure ‘Generate_ID’ secara otomatis setiap setiap kali terjadi penambahan record di tabel ‘Kasir’.

DELIMITER //

CREATE TRIGGER Before_Insert_KS

BEFORE INSERT ON Kasir

FOR EACH ROW

BEGIN

DECLARE ID_Baru VARCHAR(5);

CALL Generate_ID('KS', ID_Baru);

SET NEW.ID_Kasir = ID_Baru;

END //

DELIMITER ;

```

197  DELIMITER //
198
199  CREATE TRIGGER Before_Insert_KS
200  BEFORE INSERT ON Kasir
201  FOR EACH ROW
202  BEGIN
203      DECLARE ID_Baru VARCHAR(5);
204
205      CALL Generate_ID('KS', ID_Baru);
206
207      SET NEW.ID_Kasir = ID_Baru;
208  END //
209
210  DELIMITER ;

```

7. Membuat trigger ‘Before_Insert_OB’ untuk memanggil procedure ‘Generate_ID’ secara otomatis setiap setiap kali terjadi penambahan record di tabel ‘Obat’.

DELIMITER //

```
CREATE TRIGGER Before_Insert_OB
BEFORE INSERT ON Obat
FOR EACH ROW
BEGIN
    DECLARE ID_Baru VARCHAR(5);

    CALL Generate_ID('OB', ID_Baru);

    SET NEW.ID_Obat = ID_Baru;
END //
```

DELIMITER ;

```
212  DELIMITER //
213
214  CREATE TRIGGER Before_Insert_OB
215  BEFORE INSERT ON Obat
216  FOR EACH ROW
217  BEGIN
218      DECLARE ID_Baru VARCHAR(5);
219
220      CALL Generate_ID('OB', ID_Baru);
221
222      SET NEW.ID_Obat = ID_Baru;
223  END //
224
225  DELIMITER ;
```

8. Membuat trigger ‘Before_Insert_RE’ untuk memanggil procedure ‘Generate_ID’ secara otomatis setiap setiap kali terjadi penambahan record di tabel ‘Resep’.

DELIMITER //

```
CREATE TRIGGER Before_Insert_RE
```

```

BEFORE INSERT ON Resep
FOR EACH ROW
BEGIN
    DECLARE ID_Baru VARCHAR(5);

    CALL Generate_ID('RE', ID_Baru);

    SET NEW.ID_Resep = ID_Baru;
END //

```

DELIMITER ;

```

227  DELIMITER //
228
229  CREATE TRIGGER Before_Insert_RE
230  BEFORE INSERT ON Resep
231  FOR EACH ROW
232  BEGIN
233      DECLARE ID_Baru VARCHAR(5);
234
235      CALL Generate_ID('RE', ID_Baru);
236
237      SET NEW.ID_Resep = ID_Baru;
238  END //
239
240  DELIMITER ;

```

9. Membuat trigger ‘Before_Insert_PM’ untuk memanggil procedure ‘Generate_ID’ secara otomatis setiap setiap kali terjadi penambahan record di tabel ‘Pembayaran’.

DELIMITER //

```

CREATE TRIGGER Before_Insert_PM
BEFORE INSERT ON Pembayaran
FOR EACH ROW
BEGIN
    DECLARE ID_Baru VARCHAR(5);

```

```

    CALL Generate_ID('PM', ID_Baru);

```

```
SET NEW.ID_Pembayaran = ID_Baru;  
END //
```

```
DELIMITER ;
```

```
242  DELIMITER //  
243  
244  CREATE TRIGGER Before_Insert_PM  
245  BEFORE INSERT ON Pembayaran  
246  FOR EACH ROW  
247  BEGIN  
248      DECLARE ID_Baru VARCHAR(5);  
249  
250      CALL Generate_ID('PM', ID_Baru);  
251  
252      SET NEW.ID_Pembayaran = ID_Baru;  
253  END //  
254  
255  DELIMITER ;
```

- 10. Membuat trigger ‘Before_Insert_LG’ untuk memanggil procedure ‘Generate_ID’ secara otomatis setiap setiap kali terjadi penambahan record di tabel ‘Log’.**

```
DELIMITER //
```

```
CREATE TRIGGER Before_Insert_LG
```

```
BEFORE INSERT ON Log
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    DECLARE ID_Baru VARCHAR(5);
```

```
    CALL Generate_ID('LG', ID_Baru);
```

```
    SET NEW.ID_Log = ID_Baru;
```

```
END //
```

```
DELIMITER ;
```

```
257  DELIMITER //
258
259  CREATE TRIGGER Before_Insert_LG
260  BEFORE INSERT ON Log
261  FOR EACH ROW
262  BEGIN
263      DECLARE ID_Baru VARCHAR(5);
264
265      CALL Generate_ID('LG', ID_Baru);
266
267      SET NEW.ID_Log = ID_Baru;
268  END //
269
270  DELIMITER ;
```

- 11. Membuat trigger ‘Tambah_Dokter_Log’ untuk mencatat kejadian ke dalam tabel ‘Log’ setelah penambahan record di tabel ‘Dokter’.**

```
CREATE TRIGGER Tambah_Dokter_Log
AFTER INSERT ON Dokter
FOR EACH ROW
INSERT INTO Log(Kejadian) VALUES ('Tambah dokter baru');
```

```
272  -- Trigger untuk Log setiap Kejadian
273  CREATE TRIGGER Tambah_Dokter_Log
274  AFTER INSERT ON Dokter
275  FOR EACH ROW
276  INSERT INTO Log(Kejadian) VALUES ('Tambah dokter baru');
```

- 12. Membuat trigger ‘Tambah_Administrasi_Log’ untuk mencatat kejadian ke dalam tabel ‘Log’ setelah penambahan record di tabel ‘Administrasi’.**

```
CREATE TRIGGER Tambah_Administrasi_Log
AFTER INSERT ON Administrasi
FOR EACH ROW
INSERT INTO Log(Kejadian) VALUES ('Tambah administrasi baru');
```

```
278  CREATE TRIGGER Tambah_Administrasi_Log
279  AFTER INSERT ON Administrasi
280  FOR EACH ROW
281  INSERT INTO Log(Kejadian) VALUES ('Tambah administrasi baru');
```

- 13. Membuat trigger ‘Tambah_Pasien_Log’ untuk mencatat kejadian ke dalam tabel ‘Log’ setelah penambahan record di tabel ‘Pasien’.**

```
CREATE TRIGGER Tambah_Pasien_Log
AFTER INSERT ON Pasien
FOR EACH ROW
```

```
INSERT INTO Log(Kejadian) VALUES ('Tambah pasien baru');
```

```
283  CREATE TRIGGER Tambah_Pasien_Log  
284  AFTER INSERT ON Pasien  
285  FOR EACH ROW  
286  INSERT INTO Log(Kejadian) VALUES ('Tambah pasien baru');
```

- 14. Membuat trigger ‘Tambah_Apoteker_Log’ untuk mencatat kejadian ke dalam tabel ‘Log’ setelah penambahan record di tabel ‘Apoteker’.**

```
CREATE TRIGGER Tambah_Apoteker_Log  
AFTER INSERT ON Apoteker  
FOR EACH ROW  
INSERT INTO Log(Kejadian) VALUES ('Tambah apoteker baru');
```

```
288  CREATE TRIGGER Tambah_Apoteker_Log  
289  AFTER INSERT ON Apoteker  
290  FOR EACH ROW  
291  INSERT INTO Log(Kejadian) VALUES ('Tambah apoteker baru');
```

- 15. Membuat trigger ‘Tambah_Kasir_Log’ untuk mencatat kejadian ke dalam tabel ‘Log’ setelah penambahan record di tabel ‘Kasir’.**

```
CREATE TRIGGER Tambah_Kasir_Log  
AFTER INSERT ON Kasir  
FOR EACH ROW  
INSERT INTO Log(Kejadian) VALUES ('Tambah kasir baru');
```

```
293  CREATE TRIGGER Tambah_Kasir_Log  
294  AFTER INSERT ON Kasir  
295  FOR EACH ROW  
296  INSERT INTO Log(Kejadian) VALUES ('Tambah kasir baru');
```

- 16. Membuat trigger ‘Tambah_Obat_Log’ untuk mencatat kejadian ke dalam tabel ‘Log’ setelah penambahan record di tabel ‘Obat’.**

```
CREATE TRIGGER Tambah_Obat_Log  
AFTER INSERT ON Obat  
FOR EACH ROW  
INSERT INTO Log(Kejadian) VALUES ('Tambah obat baru');
```

```
298  CREATE TRIGGER Tambah_Obat_Log  
299  AFTER INSERT ON Obat  
300  FOR EACH ROW  
301  INSERT INTO Log(Kejadian) VALUES ('Tambah obat baru');
```

- 17. Membuat trigger ‘Ubah_Obat_Log’ untuk mencatat kejadian ke dalam tabel ‘Log’ setelah perubahan record di tabel ‘Obat’.**

```
CREATE TRIGGER Ubah_Obat_Log  
AFTER UPDATE ON Obat  
FOR EACH ROW  
INSERT INTO Log(Kejadian) VALUES ('Ubah data obat');
```

```
303  CREATE TRIGGER Ubah_Obat_Log  
304  AFTER UPDATE ON Obat  
305  FOR EACH ROW  
306  INSERT INTO Log(Kejadian) VALUES ('Ubah data obat');
```

- 18. Membuat trigger ‘Buat_Resep_Log’ untuk mencatat kejadian ke dalam tabel ‘Log’ setelah penambahan record di tabel ‘Resep’.**

```
CREATE TRIGGER Buat_Resep_Log  
AFTER INSERT ON Resep  
FOR EACH ROW  
INSERT INTO Log(Kejadian) VALUES ('Tambah resep baru');
```

```
308  CREATE TRIGGER Buat_Resep_Log  
309  AFTER INSERT ON Resep  
310  FOR EACH ROW  
311  INSERT INTO Log(Kejadian) VALUES ('Tambah resep baru');
```

- 19. Membuat trigger ‘Resep_Obat_Log’ untuk mencatat kejadian ke dalam tabel ‘Log’ setelah penambahan record di tabel ‘Resep_Obat’.**

```
CREATE TRIGGER Resep_Obat_Log  
AFTER INSERT ON Resep_Obat  
FOR EACH ROW  
INSERT INTO Log(Kejadian) VALUES ('Tambah resep obat');
```

```
313  CREATE TRIGGER Resep_Obat_Log  
314  AFTER INSERT ON Resep_Obat  
315  FOR EACH ROW  
316  INSERT INTO Log(Kejadian) VALUES ('Tambah resep obat');
```

C. Data Control Language (DCL)

1. Masuk CMD dan menggunakan MariaDB.

```
mysql -u root -P 3308 -p
```

Enter password:

```
C:\xampp\mysql\bin>mysql -u root -P 3308 -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.4.32-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

2. Meminta Seluruh Databases.

```
show databases;
```

```
MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| klinik |
| latihanweek9 |
| mysql |
| performance_schema |
| phpmyadmin |
| sistem_perpustakaan |
| toko_beli_seru |
| toko_lukas |
| universitas |
| universities_imp |
+-----+
11 rows in set (0.028 sec)
```

3. Menggunakan database klinik dan menunjukkan seluruh tabel di dalamnya.

```
MariaDB [(none)]> use klinik;
```

Database changed

```
MariaDB [klinik]> show tables;
```

```

MariaDB [(none)]> use klinik;
Database changed
MariaDB [klinik]> show tables;
+-----+
| Tables_in_klinik |
+-----+
| administrasi
| apoteker
| dokter
| id_tracker
| kasir
| kepala_klinik
| log
| obat
| pasien
| pembayaran
| resep
| resep_obat
+-----+
12 rows in set (0.002 sec)

```

4. Membuat user 1 dan memberikan akses penuh.

```

CREATE USER 'Arif'@'localhost' IDENTIFIED BY 'kepala';
GRANT SELECT, INSERT, UPDATE, DELETE ON Klinik.* TO
'Arif'@'localhost';

MariaDB [klinik]> create user 'Arif'@'localhost' IDENTIFIED BY 'kepala';
Query OK, 0 rows affected (0.022 sec)

MariaDB [klinik]> GRANT SELECT, INSERT, UPDATE, DELETE ON klinik.* TO 'Arif'@'localhost';
ERROR 1030 (HY000): Got error 176 "Read page with wrong checksum" from storage engine Aria
MariaDB [klinik]> GRANT SELECT, INSERT, UPDATE, DELETE ON klinik.* TO 'Arif'@'localhost';
Query OK, 0 rows affected (0.002 sec)

```

5. Membuat user lain (user 2) serta memberikan akses select dan insert saja.

```

CREATE USER 'Chandra'@'localhost' IDENTIFIED BY 'dokter';
GRANT SELECT, INSERT ON Klinik.* TO 'Arif'@'localhost';

MariaDB [klinik]> create user 'Chandra'@'localhost' IDENTIFIED BY 'dokter';
Query OK, 0 rows affected (0.002 sec)

MariaDB [klinik]> GRANT SELECT, INSERT ON klinik.* TO 'Arif'@'localhost';
Query OK, 0 rows affected (0.002 sec)

```

6. Menggunakan user 1 dan menjalankan tugas dari fungsi yang telah diberikan.

mysql -u Arif -P 3308 -p
Enter password: 'kepala'

```
C:\xampp\mysql\bin>mysql -u Arif -P 3308 -p
Enter password: *****
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 11
Server version: 10.4.32-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

show databases;

use klinik;

show tables;

```
MariaDB [(none)]> show databases;
+-----+
| Database      |
+-----+
| information_schema |
| klinik         |
+-----+
2 rows in set (0.001 sec)

MariaDB [(none)]> use klinik;
Database changed
MariaDB [klinik]> show tables;
+-----+
| Tables_in_klinik |
+-----+
| administrasi    |
| apoteker        |
| dokter          |
| id_tracker      |
| kasir           |
| kepala_klinik   |
| log             |
| obat            |
| pasien          |
| pembayaran      |
| resep           |
| resep_obat     |
+-----+
```

a. menjalankan fungsi SELECT

SELECT * FROM obat;

```
MariaDB [klinik]> SELECT * FROM obat;
+-----+-----+-----+-----+
| ID_Obat | Nama_Obat | Harga   | Stok   |
+-----+-----+-----+-----+
| OB001  | Paracetamol | 5000.00 | 100   |
| OB002  | Amoxicillin | 10000.00 | 200   |
| OB003  | Ibuprofen   | 7500.00 | 150   |
| OB004  | Vitamin C    | 3000.00 | 250   |
| OB005  | Cetirizine   | 5000.00 | 100   |
| OB006  | Antasida    | 2000.00 | 300   |
| OB007  | Loratadine   | 6000.00 | 180   |
| OB008  | Omeprazole   | 8000.00 | 120   |
| OB009  | Metformin    | 10000.00 | 100   |
| OB010  | Amlodipine   | 9000.00 | 140   |
| OB011  | Promaag     | 10000.00 | 170   |
+-----+-----+-----+-----+
11 rows in set (0.023 sec)
```

b. Menjalankan fungsi INSERT

```
INSERT INTO obat (ID_Obat, Nama_Obat, Harga, Stok)
VALUES ('OB012', 'Vitamin D', 5000, 200);
```

```
MariaDB [klinik]> INSERT INTO obat (ID_Obat, Nama_Obat, Harga, Stok)
Query OK, 1 row affected (0.026 sec)

MariaDB [klinik]> SELECT * FROM obat;
+----+----+----+----+
| ID_Obat | Nama_Obat | Harga | Stok |
+----+----+----+----+
| OB001 | Paracetamol | 5000.00 | 100 |
| OB002 | Amoxicillin | 10000.00 | 200 |
| OB003 | Ibuprofen | 7500.00 | 150 |
| OB004 | Vitamin C | 3000.00 | 250 |
| OB005 | Cetirizine | 5000.00 | 100 |
| OB006 | Antasida | 2000.00 | 300 |
| OB007 | Loratadine | 6000.00 | 180 |
| OB008 | Omeprazole | 8000.00 | 120 |
| OB009 | Metformin | 10000.00 | 100 |
| OB010 | Amlodipine | 9000.00 | 140 |
| OB011 | Promaag | 10000.00 | 170 |
| OB014 | Vitamin D | 5000.00 | 200 |
+----+----+----+----+
12 rows in set (0.000 sec)
```

c. Menjalankan fungsi UPDATE

```
UPDATE obat SET Nama_Obat = 'Vitamin E'
WHERE ID_Obat = 'OB014';
```

```
MariaDB [klinik]> UPDATE obat SET Nama_Obat = 'Vitamin E'
Query OK, 1 row affected (0.004 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [klinik]> SELECT * FROM obat;
+----+----+----+----+
| ID_Obat | Nama_Obat | Harga | Stok |
+----+----+----+----+
| OB001 | Paracetamol | 5000.00 | 100 |
| OB002 | Amoxicillin | 10000.00 | 200 |
| OB003 | Ibuprofen | 7500.00 | 150 |
| OB004 | Vitamin C | 3000.00 | 250 |
| OB005 | Cetirizine | 5000.00 | 100 |
| OB006 | Antasida | 2000.00 | 300 |
| OB007 | Loratadine | 6000.00 | 180 |
| OB008 | Omeprazole | 8000.00 | 120 |
| OB009 | Metformin | 10000.00 | 100 |
| OB010 | Amlodipine | 9000.00 | 140 |
| OB011 | Promaag | 10000.00 | 170 |
| OB014 | Vitamin E | 5000.00 | 200 |
+----+----+----+----+
12 rows in set (0.000 sec)
```

d. Menjalankan Fungsi DELETE

```
DELETE FROM obat WHERE ID_Obat = 'OB014';
```

```

MariaDB [klinik]> DELETE FROM obat WHERE ID_Obat = 'OB014';
Query OK, 1 row affected (0.004 sec)

MariaDB [klinik]> SELECT * FROM obat;
+-----+-----+-----+-----+
| ID_Obat | Nama_Obat | Harga | Stok |
+-----+-----+-----+-----+
| OB001   | Paracetamol | 5000.00 | 100 |
| OB002   | Amoxicillin | 10000.00 | 200 |
| OB003   | Iuprofen    | 7500.00 | 150 |
| OB004   | Vitamin C   | 3000.00 | 250 |
| OB005   | Cetirizine  | 5000.00 | 100 |
| OB006   | Antasida    | 2000.00 | 300 |
| OB007   | Loratadine  | 6000.00 | 180 |
| OB008   | Omeprazole  | 8000.00 | 120 |
| OB009   | Metformin   | 10000.00 | 100 |
| OB010   | Amlodipine  | 9000.00 | 140 |
| OB011   | Promaag     | 10000.00 | 170 |
+-----+-----+-----+-----+
11 rows in set (0.000 sec)

```

7. Menggunakan user 2 dan menjalankan fungsi yang diberikan.

mysql -u Chandra -P 3308 -p

Enter password: ‘dokter’

```

C:\xampp\mysql\bin>mysql -u Chandra -P 3308 -p
Enter password: *****
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 13
Server version: 10.4.32-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```

use klinik;

show tables;

```

MariaDB [(none)]> use klinik;
Database changed
MariaDB [klinik]> show tables;
+-----+
| Tables_in_klinik |
+-----+
| administrasi      |
| apoteker          |
| dokter            |
| id_tracker        |
| kasir             |
| kepala_klinik     |
| log               |
| obat              |
| pasien            |
| pembayaran        |
| resep             |
| resep_obat        |
+-----+
12 rows in set (0.001 sec)

```

a. Menggunakan fungsi SELECT

SELECT * FROM obat;

```
MariaDB [klinik]> select * from obat;
+-----+-----+-----+-----+
| ID_Obat | Nama_Obat | Harga | Stok |
+-----+-----+-----+-----+
| OB001   | Paracetamol | 5000.00 | 100 |
| OB002   | Amoxicillin | 10000.00 | 200 |
| OB003   | Ibuprofen   | 7500.00 | 150 |
| OB004   | Vitamin C    | 3000.00 | 250 |
| OB005   | Cetirizine   | 5000.00 | 100 |
| OB006   | Antasida     | 2000.00 | 300 |
| OB007   | Loratadine   | 6000.00 | 180 |
| OB008   | Omeprazole   | 8000.00 | 120 |
| OB009   | Metformin    | 10000.00 | 100 |
| OB010   | Amlodipine   | 9000.00 | 140 |
| OB011   | Promaag      | 10000.00 | 170 |
+-----+-----+-----+-----+
```

- b. Menggunakan fungsi INSERT

```
INSERT INTO obat (ID_Obat, Nama_Obat, Harga, Stok)
VALUES ('OB012', 'Vitamin D', 5000, 200);
```

```
MariaDB [klinik]> INSERT INTO obat (ID_Obat, Nama_Obat, Harga, Stok) VALUES ('OB012', 'Vitamin D', 5000, 200);
Query OK, 1 row affected (0.004 sec)

MariaDB [klinik]> select * from obat;
+-----+-----+-----+-----+
| ID_Obat | Nama_Obat | Harga | Stok |
+-----+-----+-----+-----+
| OB001   | Paracetamol | 5000.00 | 100 |
| OB002   | Amoxicillin | 10000.00 | 200 |
| OB003   | Ibuprofen   | 7500.00 | 150 |
| OB004   | Vitamin C    | 3000.00 | 250 |
| OB005   | Cetirizine   | 5000.00 | 100 |
| OB006   | Antasida     | 2000.00 | 300 |
| OB007   | Loratadine   | 6000.00 | 180 |
| OB008   | Omeprazole   | 8000.00 | 120 |
| OB009   | Metformin    | 10000.00 | 100 |
| OB010   | Amlodipine   | 9000.00 | 140 |
| OB011   | Promaag      | 10000.00 | 170 |
| OB012   | Vitamin D    | 5000.00 | 200 |
+-----+-----+-----+-----+
12 rows in set (0.001 sec)
```

- c. Menggunakan fungsi UPDATE

```
UPDATE obat SET Nama_Obat = 'Vitamin E'
WHERE ID_Obat = 'OB013';
```

```
MariaDB [klinik]> UPDATE obat SET Nama_Obat = 'Vitamin E' WHERE ID_Obat = 'OB013';
ERROR 1142 (42000): UPDATE command denied to user 'Chandra'@'localhost' for table 'klinik'.`obat`
MariaDB [klinik]> |
```

- d. Menggunakan fungsi DELETE

```
DELETE FROM obat WHERE ID_Obat = 'OB013';
```

```
MariaDB [klinik]> DELETE FROM obat WHERE ID_Obat = 'OB013';
ERROR 1142 (42000): DELETE command denied to user 'Chandra'@'localhost' for table 'klinik'.`obat`
MariaDB [klinik]> |
```

BAB V

LAINNYA

A. Pembagian Kelompok

No.	Nama	NIM	Tugas
1	Vanesa Rizka Alfatihah	102022300121	Relational Model, Penjelasan ERD, Stored Procedure, & Trigger
2	Zaky Aprilian	102022300212	ERD, Penjelasan Relational Model, & Power Point
3	Gyebran Nauri Haikal	102022300389	ERD, Latar Belakang, & Tujuan dan Manfaat
4	Lukas Ricky Krisjatmiko	102022330321	DDL, DML, & DCL