Modul Praktikum I
Matakuliah Basis Data Lanjut

"Penggunaan DDL dan DML dalam *Query* Basis Data"

A. TUJUAN

- Mahasiswa mampu menggunakan perintah CREATE untuk membuat objek-objek basis data.
- Mahasiswa mampu menggunakan perintah ALTER untuk memodifikasi struktur tabel.
- Mahasiswa mampu menggunakan perintah DROP untuk menghancurkan objek-objek basis data.
- Mahasiswa mampu menggunakan perintah SELECT dan variannya untuk mencari dan menampilkan data.
- Mahasiswa mampu menggunakan perintah INSERT dan variannya untuk menyimpan data ke dalam tabel.
- Mahasiswa mampu menggunakan perintah UPDATE dan variannya untuk memodifikasi data di dalam tabel.
- Mahasiswa mampu menggunakan perintah DELETE dan variannya untuk menghapus data dari dalam tabel.

B. PETUNJUK PELAKSANAAN PRAKTIKUM

- Awali setiap aktivitas dengan do'a, semoga berkah dan mendapat kemudahan.
- Pahami tujuan dan dasar teori dengan baik dan benar.
- Kerjakan latihan dan tugas-tugas praktikum dengan baik, sabar, mandiri, dan jujur.
- Tanyakan kepada asisten praktikum / dosen apabila ada hal-hal yang kurang jelas / menjumpai kesulitan saat melaksanakan kegiatan praktikum.

C. DASAR TEORI

Basis data merupakan sebuah wadah di dalam komputer yang dapat digunakan untuk menyimpan sekumpulan data. Selain itu basis data juga dapat dimanfaatkan dalam pengelolaan data, baik pencarian data, pengubahan data, bahkan penghapusan data. Oleh karena itu dibutuhkan suatu manajemen pada basis data tersebut agar pengelolaan data menjadi mudah.

Salah satu *Relational Database Management System* (RDBMS) yang terkenal ialah MySQL dan turunannya, MariaDB. RDBMS tersebut berfungsi layaknya sebuah *server* yang melayani pemrosesan data. Bahasa yang dapat digunakan untuk melakukan manajemen terhadap basis data pada RDBMS tersebut ialah *Structured Query Language* (SQL).

SQL merupakan bahasa standar yang digunakan dalam pengelolaan basis data termasuk data-data yang tersimpan di dalamnya. Bahasa ini dapat dibedakan ke dalam 2 kelompok perintah, yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML).

1. *Data Definition Language* (DDL)

DDL merupakan kelompok perintah pada SQL yang berkaitan dengan pengelolaan objek-objek basis data seperti basis data itu sendiri, tabel, *view*, indeks, *stored procedure*, *trigger*, dan lain-lain. Pada kelompok perintah ini terdapat 3 perintah utama, yaitu CREATE, ALTER, dan DROP.

Perintah CREATE berfungsi untuk menciptakan atau menginisialisasi objek basis data. Perintah ALTER berfungsi untuk memodifikasi struktur objek basis data. Sedangkan perintah DROP berfungsi untuk menghapus objek-objek basis data.

Selain perintah-perintah utama tersebut terdapat pula perintah-perintah lainnya. Salah satunya ialah perintah SHOW yang dapat digunakan untuk menampilkan daftar nama objek-objek basis data yang telah diciptakan. Ada pula perintah DESC yang dapat digunakan untuk melihat struktur tabel yang ada di dalam suatu basis data.

1.1 Menciptakan Basis Data / *Database*

Basis data atau dalam bahasa Inggris disebut *database* merupakan media utama dalam pengelolaan data. Di dalam *database* dapat terkandung sekumpulan tabel, *view*, indeks, dan berbagai macam objek basis data lainnya. Untuk membuat sebuah *database* baru di dalam MySQL digunakan perintah berikut:

```
CREATE DATABASE nama_database;
```

Saat membuat suatu *database* harus dipastikan bahwa nama *database* yang akan dibuat belum atau tidak sedang digunakan oleh *database* lainnya di dalam mesin atau komputer yang sama. Untuk memeriksa daftar nama *database* yang ada di dalam MySQL dapat menggunakan perintah berikut:

```
SHOW DATABASES;
```

Database yang telah dibuat tidak serta merta dapat langsung digunakan. Sebelum melakukan kegiatan-kegiatan pengelolaan data di dalam suatu *database*, maka pengguna harus memberikan perintah untuk masuk ke dalam *database* yang diinginkan. Bentuk perintah atau *query* untuk masuk ke dalam suatu *database* adalah sebagai berikut:

```
USE nama_database;
```

1.2 Menciptakan Tabel

Tabel merupakan objek basis data yang berfungsi sebagai media penyimpanan data. Suatu tabel terdiri dari sekumpulan kolom (*field*) dan baris (*row*). Kolom-kolom pada tabel menunjukkan jenis atau kriteria data yang disimpan. Oleh karena itu setiap kolom pada suatu tabel memiliki spesifikasi tersendiri dan harus ditentukan saat tabel diciptakan. Sedangkan baris menunjukkan sekumpulan data yang saling terkait.

Untuk menciptakan suatu tabel digunakan format *query* sebagai berikut:

```
CREATE TABLE nama_tabel(
  nama_kolom1 tipe_data_kolom1  spesifikasi_kolom1,
  nama_kolom2 tipe_data_kolom2  spesifikasi_kolom2,
  ...
  nama_kolomX tipe_data_kolomX  spesifikasi_kolomX
);
```

Sama dengan saat membuat sebuah *database*, saat membuat atau menciptakan sebuah tabel harus dipastikan terlebih dahulu nama tabel yang akan dibuat belum atau tidak sedang digunakan oleh tabel lainnya di dalam *database* yang sama. Untuk melihat daftar nama tabel yang ada di dalam *database* dapat dilakukan dengan menggunakan perintah berikut:

```
SHOW TABLES;
```

Saat menciptakan suatu tabel, masing-masing kolom pada tabel tersebut juga didefinisikan lengkap dengan tipe data dan spesifikasi-spesifikasi tambahan lainnya. Terdapat berbagai macam tipe data kolom untuk suatu tabel di dalam MySQL. Tipe-tipe data tersebut ditunjukkan pada Tabel 1 hingga Tabel 3.

Tabel 1. Tipe Data Numeris

Tipe Data	Keterangan	Memori (byte)
BIT(*M*)	Untuk menyimpan bilangan *M* buah bilangan biner (0 atau 1), dimana M bernilai 1 – 64.	(*M*/7)/8
TINYINT(*M*)	Untuk menyimpan bilangan bulat antara -128 hingga 127 atau 0 hingga 255 dalam kondisi *unsigned*.	1
BOOL BOOLEAN	Untuk menyatakan nilai benar (TRUE) atau salah (FALSE).	1
SMALLINT(*M*)	Untuk menyimpan bilangan bulat antara -32768 hingga 32767 atau antara 0 hingga 65535 dalam kondisi *unsigned*.	2
MEDIUMINT(*M*)	Untuk menyimpan bilangan bulat antara -8388608 hingga 8388607 atau 0 hingga 16777215 dalam kondisi *unsigned*.	3
INT(*M*) INTEGER(*M*)	Untuk menyimpan bilangan bulat antara -2147683648 hingga2147683647 atau 0 hingga 4294967295 dalam kondisi *unsigned*.	4
BIGINT(*M*)	Untuk menyimpan bilangan bulat antara -9223372036854775808 hingga 9223372036854775807 atau 0 hingga 18446744073709551615 dalam kondisi *unsigned*.	6
FLOAT(*M*,*D*) FLOAT(*P*)	Untuk menyimpan bilangan pecahan antara $-3{,}402823466 \times 10^{38}$ hingga $3{,}402823466 \times 10^{38}$.	4
DOUBLE(*M*,*D*) PRECISION(*M*,*D*)	Untuk menyimpan bilangan pecahan antara $-1{,}7976931348623157 \times 10^{308}$ hingga $1{,}7976931348623157 \times 10^{308}$.	8
DEC(*M*,*D*) DECIMAL(*M*,*D*)	Untuk menyimpan bilangan pecahan yang memerlukan keakurasian tinggi.	

Keterangan:

- *M* pada tipe data TINYINT, SMALLINT, MEDIUMINT, INT, dan BIGINT menyatakan lebar *display* data
- *M* pada tipe data FLOAT, DOUBLE, dan DECIMAL menyatakan jumlah digit desimal
- *D* pada tipe data FLOAT, DOUBLE, dan DECIMAL menyatakan jumlah digit di belakang tanda desimal
- *P* pada tipe data FLOAT menyatakan jumlah presisi data dalam bit

Tabel 2. Tipe Data Tanggal dan Waktu

Tipe Data	Keterangan	Memori (byte)
DATE	Untuk data tanggal antara 1 Januari 1000 hingga 31 Desember 9999. Format: 'yyyy-mm-dd' | y = tahun, m = bulan, d = tanggal	3
TIME	Untuk data waktu. Format: 'hh:mm:ss' | h = jam, m = menit, s = detik	3
DATETIME	Untuk menyimpan data tanggal sekaligus waktu. Format: 'yyyy-mm-dd hh:mm:ss'	8
TIMESTAMP	Untuk menyimpan data tanggal sekaligus waktu, dimana datanya akan diisi secara otomatis saat terjadi operasi INSERT atau UPDATE.	4
YEAR(*R*)	Untuk menyimpan data tahun dengan format *R* digit, dimana *R* dapat bernilai 2 atau 4.	1

Tabel 3. Tipe Data String

Tipe Data	Keterangan	Memori (byte)
CHAR(*M*)	Untuk menyimpan data yang terdiri dari maksimal *M* buah karakter. Jika data yang disimpan kurang dari *M* buah karakter, maka sisanya akan diisi dengan spasi. Nilai *M* maksimal adalah 255.	*M*
VARCHAR(*M*)	Untuk menyimpan data string dengan *L* buah karakter (maksimal *M* buah karakter). Nilai *M* maksimal adalah 65535.	$L+1, L \leq M$
BINARY(*M*)	Serupa dengan CHAR(*M*), namun disimpan dalam bentuk biner.	*M*
VARBINARY(*M*)	Serupa dengan VARCHAR(*M*), namun disimpan dalam bentuk biner.	$L+1, L \leq M$
TINYBLOB BLOB MEDIUMBLOB LONGBLOB	Untuk menyimpan data string dalam format biner (tanpa himpunan karakter, pengurutan dan pembandingan dilakukan berdasarkan nilai numerik dari bit-bit yang merepresentasikan string).	$L+1, L < 2^8$ $L+2, L < 2^{16}$ $L+3, L < 2^{24}$ $L+4, L < 2^{32}$
TINYTEXT TEXT MEDIUMTEXT LONGTEXT	Untuk menyimpan data string dalam format karakter (memiliki himpunan karakter, pengurutan dan pembandingan dilakukan berdasarkan perbandingan pada himpunan karakter).	$L+1, L < 2^8$ $L+2, L < 2^{16}$ $L+3, L < 2^{24}$ $L+4, L < 2^{32}$
ENUM('nilai 1', nilai 2', ...)	Untuk menyimpan data dengan pilihan-pilihan yang telah ditentukan (nilai 1, nilai 2, dst). Biasanya digunakan untuk data-data dengan pilihan terbatas seperti jenis kelamin, pendidikan, dsb. Jumlah elemen atau pilihan data pada ENUM dapat mencapai 65535 elemen.	1 atau 2
SET('nilai 1', 'nilai 2', ...)	Untuk menyimpan data himpunan dari sekumpulan pilihan. Tipe data ini mirip dengan ENUM, hanya saja dapat menyimpan nol, satu, atau lebih kombinasi data dari pilihan-pilihan pada elemen SET.	1 – 8

Spesifikasi atau atribut tambahan untuk kolom saat menciptakan sebuah tabel akan menentukan sifat dari kolom tersebut. Apabila suatu kolom akan dijadikan sebagai *primary key* tabel terkait, maka atribut PRIMARY KEY harus ditambahkan pada kolom tersebut saat pendefinisiannya. Apabila suatu kolom harus diisi saat terjadi penyimpanan data, maka atribut NOT NULL harus ditambahkan saat pendefinisiannya. Apabila data pada suatu kolom memiliki

nilai standar (*default*), maka atribut DEFAULT *nilai_default* harus ditambahkan saat pendefinisian kolom tersebut. Apabila diinginkan agar nilai suatu kolom yang menjadi *primary key* secara otomatis dapat bertambah setiap kali ada data baru yang dimasukkan, maka atribut AUTO_INCREMENT harus ditambahkan saat pendefinisian kolom tersebut. Apabila diinginkan agar nilai pada suatu kolom bersifat unik (tidak ada data kembar), maka atribut UNIQUE harus ditambahkan saat pendefinisiannya. Untuk melihat definisi struktur suatu tabel dapat dilakukan dengan menggunakan perintah berikut:

```
DESC nama_tabel;
```

1.3 Memodifikasi Struktur Tabel

Tabel yang telah dibuat pada suatu *database* mungkin saja harus diubah atau dimodifikasi strukturnya, seperti nama tabelnya, nama kolomnya, hingga spesifikasi kolom-kolom di dalam tabel terkait. Secara umum untuk melakukan perubahan-perubahan tersebut dapat dilakukan dengan memanfaatkan perintah ALTER.

Untuk mengubah nama suatu tabel dapat dilakukan dengan menggunakan perintah berikut:

```
ALTER TABLE nama_tabel_lama
RENAME nama_table_baru;
```

Atau dapat disederhanakan menjadi:

```
RENAME nama_tabel_lama TO nama_tabel_baru;
```

Untuk mengubah nama kolom pada suatu tabel dapat dilakukan dengan menggunakan perintah berikut:

```
ALTER TABLE nama_tabel
CHANGE nama_kolom_lama nama_kolom_baru definisi_kolom_baru;
```

Untuk mengubah definisi atau spesifikasi suatu kolom dapat dilakukan dengan menggunakan perintah berikut:

```
ALTER TABLE nama_tabel
MODIFY nama_kolom definisi_baru;
```

Untuk menambahkan kolom baru pada suatu tabel dapat dilakukan dengan menggunakan perintah berikut:

```
ALTER TABLE nama_tabel
ADD nama_kolom_baru definisi_kolom_baru [FIRST | AFTER
nama_kolom];
```

Atribut FIRST dan AFTER (tanpa tanda kurung siku) bersifat opsional. Atribut FIRST menandakan bahwa kolom baru yang ditambahkan akan diletakkan sebagai kolom pertama pada tabel, sedangkan atribut AFTER menandakan kolom baru yang ditambahkan akan diletakkan di kanan kolom lainnya.

Untuk menghapus suatu kolom dapat dilakukan dengan menggunakan perintah berikut:

```
ALTER TABLE nama_tabel
DROP nama_kolom;
```

1.4 Menghapus Tabel

Tabel yang telah dibuat pada suatu *database* mungkin saja tidak dibutuhkan lagi. Oleh karena itu suatu tabel dapat dihapus atau dihancurkan. Untuk menghapus suatu tabel dari dalam *database* dapat dilakukan dengan menggunakan perintah berikut:

```
DROP TABLE nama_tabel;
```

Apabila di dalam tabel masih terdapat sekumpulan data, maka data-data tersebut secara otomatis juga akan terhapus saat tabel tersebut dihancurkan.

1.5 Menghapus Basis Data / *Database*

Selain tabel, sebuah *database* juga dapat dihapus atau dihancurkan dari dalam mesin / komputer. Untuk menghapus sebuah database dapat dilakukan dengan menggunakan perintah berikut:

```
DROP DATABASE nama_database;
```

Penghapusan terhadap suatu *database* akan mengakibatkan objek-objek serta data-data di dalamnya turut terhapus secara permanen.

2. *Data Manipulation Language* (DML)

DML merupakan kelompok perintah pada SQL yang berkaitan dengan pengelolaan data-data yang tersimpan di dalam basis data. Pengelolaan data tersebut dapat berupa penyimpanan data baru, pengubahan data, penghapusan data, hingga pencarian data dengan kriteria tertentu. Perintah utama pada kelompok DML terdiri dari 4 perintah, yaitu SELECT, INSERT, UPDATE, dan DELETE.

Perintah SELECT berfungsi untuk menampilkan atau mencari data dari sekumpulan tabel di dalam basis data. Perintah INSERT berfungsi untuk menyimpan data ke dalam suatu tabel. Perintah UPDATE berfungsi untuk mengubah data yang telah di simpan di dalam tabel. Sedangkan perintah DELETE berfungsi untuk menghapus data dari dalam tabel. Masing-masing perintah tersebut memiliki variasi bentuk yang dapat digunakan untuk berbagai macam kondisi/sesuai dengan kebutuhan pengguna.

2.1 Menyimpan Data

Setelah tabel dibuat di dalam *database*, maka tabel tersebut dapat digunakan untuk menyimpan data. Penyimpanan data ke dalam suatu tabel dilakukan dengan menggunakan perintah berikut:

```
INSERT INTO nama_tabel
VALUES ('data 1', 'data 2', ...);
```

Perintah tersebut akan menyimpan satu baris data ke dalam tabel. Urutan data yang dituliskan di dalam tanda kurung klausa VALUES harus sesuai dengan urutan kolom pada tabel yang dipilih. Begitu pula dengan jumlah datanya, harus sama dengan jumlah kolom pada tabel tujuan.

Apabila jumlah baris data yang akan disimpan lebih dari satu baris, maka format perintah yang digunakan adalah sebagai berikut:

```
INSERT INTO nama_tabel
VALUES
('baris 1 data 1', 'baris 1 data 2', ...),
('baris 2 data 1', 'baris 2 data 2', ...),
...
('baris X data 1', 'baris X data 2', ...);
```

Dengan menggunakan perintah di atas, maka *X* buah baris dapat disimpan sekaligus ke dalam tabel tujuan.

Apabila data yang dimasukkan tidak mengisi seluruh kolom yang ada pada tabel tujuan, maka format perintah yang digunakan adalah sebagai berikut:

```
INSERT INTO nama_tabel(nama_kolom1, nama_kolom2, ...)
VALUES ('data untuk kolom 1', 'data untuk kolom 2', ...);
```

Perlu diperhatikan bahwa kolom yang boleh tidak diisi adalah kolom-kolom yang saat pendefinisiannya tidak diberikan atribut NOT NULL dan bukan *primary key*. Untuk kolom-kolom tersebut, maka akan muncul keterangan *null* yang menandakan bahwa tidak terdapat data pada kolom terkait atau akan muncul nilai *default* jika kolom tersebut memberlakukan nilai standar saat pendefinisiannya.

2.2 Mengubah Data

Data yang telah disimpan di dalam tabel mungkin saja perlu diubah karena ada kesalahan atau karena perlu dilakukan pembaharuan. Untuk mengubah data di dalam tabel dapat dilakukan dengan menggunakan perintah berikut:

```
UPDATE nama_tabel
SET    nama_kolom1 = 'data baru untuk kolom 1',
       nama_kolom2 = 'data baru untuk kolom 2',
       ...
       nama_kolomX = 'data baru untuk kolom X'
WHERE kriteria_data;
```

Pada perintah tersebut penggunaan klausa WHERE bersifat opsional. Penggunaan klausa WHERE akan menyebabkan perubahan data hanya terjadi pada baris data yang memenuhi kriteria. Akan tetapi apabila klausa WHERE tidak digunakan, maka perubahan data akan terjadi pada seluruh baris data yang ada di dalam tabel.

2.3 Menghapus Data

Data yang ada di dalam tabel dapat dihapus apabila sudah tidak dibutuhkan lagi. Untuk menghapus data dari dalam tabel digunakan perintah berikut:

```
DELETE FROM nama_tabel
WHERE kriteria_data;
```

Seperti pada perintah untuk pengubahan data, penggunaan klausa WHERE pada perintah untuk penghapusan data juga bersifat opsional. Apabila klausa tersebut digunakan, maka data yang dihapus hanyalah data yang sesuai dengan kriteria. Namun apabila klausa tersebut tidak digunakan, maka seluruh data yang ada di dalam tabel akan terhapus.

2.4 Mencari /Menampilkan Data

Sekumpulan data di dalam tabel dapat ditampilkan seluruhnya atau sebagian saja tergantung pada kebutuhan. Untuk menampilkan seluruh data yang ada di dalam suatu tabel dapat dilakukan dengan menggunakan perintah berikut:

```
SELECT * FROM nama_tabel;
```

Operator * pada perintah tersebut bermakna “seluruh kolom”.

Untuk menampilkan data hanya dari kolom tertentu, maka format perintah yang digunakan adalah sebagai berikut:

```
SELECT nama_kolom1, nama_kolom2, ...
FROM nama_tabel;
```

Perintah tersebut akan menampilkan seluruh baris data namun terbatas hanya pada kolom-kolom yang namanya dituliskan pada bagian SELECT.

Terkadang data yang ingin ditampilkan hanyalah data-data yang sesuai dengan suatu kondisi atau kriteria. Untuk kasus-kasus seperti itu, maka klausa WHERE dapat ditambahkan ke dalam perintah SELECT seperti berikut:

```
SELECT [nama_kolom | *]
FROM nama_tabel
WHERE kriteria;
```

Kriteria atau kondisi suatu data pada kalusa WHERE dapat melibatkan berbagai macam operator, seperti operator relasional (=, <, >, <=, >=, <>), boolean (AND, OR, XOR, NOT), IS, dan LIKE.

Apabila diinginkan hasil pencarian data diurutkan berdasarkan kolom tertentu, maka pada perintah SELECT dapat ditambahkan klausa ORDER BY seperti berikut:

```
SELECT [nama_kolom | *]
FROM nama_tabel
ORDER BY nama_kolom [DESC];
```

Penggunaan atribut DESC (tanpa tanda kurung siku) pada perintah tersebut bersifat opsional. Apabila atribut tersbeut digunakan, maka data akan diurutkan secara *descending*.

Pada hasil pencarian sekumpulan data terkadang ditemukan data kembar atau data yang muncul lebih dari satu kali. Untuk mengeliminasi duplikasi data pada hasil pencarian dapat dilakukan dengan melibatkan atribut DISTINCT pada perintah SELECT seperti berikut:

```
SELECT DISTINCT [nama_kolom | *]
FROM nama_tabel;
```

D. LATIHAN

Pada latihan ini akan dibuat sebuah *database* bernama **kampus** yang di dalamnya terdapat tabel-tabel berikut:

Tabel: **mahasiswa**

Nama Kolom	Tipe Data	Panjang/Elemen	Spesifikasi
nim	CHAR	3	Primary key
nama_mahasiswa	VARCHAR	50	Not null
jenis_kelamin	ENUM	L, P	Not null
tempat_lahir	VARCHAR	15	Not null
tanggal_lahir	DATE	-	Not null
alamat	TEXT	-	Not null
semester	INT	2	Default = 1
jurusan	ENUM	D3 MI, S1 TI	Not null

Tabel: **dosen**

Nama Kolom	Tipe Data	Panjang/Elemen	Spesifikasi
id_dosen	CHAR	3	Primary key
nama_dosen	VARCHAR	50	Not null
jenis_kelamin	ENUM	L, P	Not null

Tabel: **matakuliah**

Nama Kolom	Tipe Data	Panjang/Elemen	Spesifikasi
kode_matakuliah	CHAR	3	Primary key
nama_matakuliah	VARCHAR	30	Not null, unique
sks	INT	1	Not null
semester	ENUM	(‘Ganjil’,’Genap’)	Not null

Tabel: **kelas**

Nama Kolom	Tipe Data	Panjang/Elemen	Spesifikasi
kode_kelas	INT	-	Primary key, auto increment
kode_matakuliah	CHAR	3	Not null
id_dosen	CHAR	3	-

Tabel: **peserta_kuliah**

Nama Kolom	Tipe Data	Panjang/Elemen	Spesifikasi
kode_peserta	INT	-	Primary key, auto increment
kode_kelas	INT	-	Not null, unique
nim	CHAR	3	Not null, unique
waktu	TIMESTAMP	-	-

Keterangan: seorang mahasiswa tidak bisa mengikuti kelas yang sama lebih dari 1 kali

1. Membuat Basis Data

Untuk membuat *database* bernama **kampus**, jalankan *query* berikut:

```
CREATE DATABASE kampus;
```

Agar *database* tersebut dapat digunakan, maka berikan perintah untuk masuk/menggunakan *database* tersebut dengan menjalankan *query* berikut:

```
USE kampus;
```

2. Membuat Tabel

Untuk membuat tabel **mahasiswa** jalankan *query* berikut:

```
CREATE TABLE mahasiswa(
  nim               CHAR(3)                 PRIMARY KEY,
  nama_mahasiswa    VARCHAR(50)             NOT NULL,
  jenis_kelamin     ENUM('L','P')           NOT NULL,
  tempat_lahir      VARCHAR(15)             NOT NULL,
  tanggal_lahir     DATE                    NOT NULL,
  alamat            TEXT                    NOT NULL,
  semester          INT(2)                  DEFAULT 1,
  jurusan           ENUM('D3 MI','S1 TI')   NOT NULL
);
```

Untuk membuat tabel **dosen** jalankan *query* berikut:

```
CREATE TABLE dosen(
  id_dosen              CHAR(3)                 PRIMARY KEY,
  nama_dosen            VARCHAR(50)             NOT NULL,
  jenis_kelamin         ENUM('L','P')           NOT NULL
);
```

Untuk membuat tabel **matakuliah** jalankan *query* berikut:

```
CREATE TABLE matakuliah(
  kode_matakuliah   CHAR(3)                 PRIMARY KEY,
  nama_matakuliah   VARCHAR(30)             NOT NULL UNIQUE,
  sks               INT(1)                  NOT NULL,
  semester          ENUM('Ganjil','Genap')  NOT NULL
);
```

Untuk membuat tabel **kelas** jalankan *query* berikut:

```
CREATE TABLE kelas(
  kode_kelas         INT          PRIMARY KEY AUTO_INCREMENT,
  kode_matakuliah    CHAR(3)      NOT NULL,
  id_dosen           CHAR(3)
);
```

Untuk membuat tabel **peserta_kuliah** jalankan *query* berikut:

```
CREATE TABLE peserta_kuliah(
  kode_peserta       INT          PRIMARY KEY AUTO_INCREMENT,
  kode_kelas         INT          NOT NULL,
  nim                CHAR(3)      NOT NULL,
  waktu              TIMESTAMP,
  UNIQUE KEY(kode_kelas, nim)
);
```

Untuk memastikan kesesuaian struktur tabel yang telah dibuat dengan yang diminta, maka dapat menjalankan masing-masing *query* berikut untuk tiap tabel:

```
DESC mahasiswa;
DESC dosen;
DESC matakuliah;
DESC kelas;
DESC peserta_kuliah;
```

Perhatikan kolom-kolom yang memiliki spesifikasi *auto increment* maupun *default*! Apakah ada perbedaan informasi dengan kolom-kolom lainnya? Perhatikan pula kolom-kolom yang memiliki tipe data TIMESTAMP! Apakah secara otomatis sifatnya adalah tidak boleh kosong (*not null*)? Apa nilai *default* yang diberikan oleh sistem untuk kolom bertipe TIMESTAMP tersebut?

3. Memodifikasi Struktur Tabel

Anggaplah akan dilakukan beberapa modifikasi terhadap struktur dari tabel-tabel yang telah dibuat sebelumnya. Sebagai contoh apabila ingin menambahkan sebuah kolom baru bernama **pendidikan_terakhir** pada tabel **dosen** dengan tipe data **ENUM** dimana elemen-elemennya adalah **S1**, **S2**, dan **S3** dan kolom tersebut wajib diisi, maka *query* untuk kasus tersebut adalah sebagai berikut:

```
ALTER TABLE dosen
ADD pendidikan_terakhir ENUM('S1','S2','S3') NOT NULL;
```

Contoh lainnya adalah apabila diinginkan agar kolom **sks** pada tabel **matakuliah** mempunyai nilai ***default* 2**, maka *query* untuk mengubah spesifikasi kolom tersebut adalah sebagai berikut:

```
ALTER TABLE matakuliah
MODIFY sks INT(1) DEFAULT 2;
```

Apabila diinginkan kolom **waktu** pada tabel **peserta_kuliah** diubah namanya menjadi **waktu_daftar**, maka *query* untuk kasus tersebut adalah sebagai berikut:

```
ALTER TABLE peserta_kuliah
CHANGE waktu waktu_daftar TIMESTAMP;
```

Apabila kolom **semester** pada tabel **matakuliah** tidak dibutuhkan karena dianggap setiap matakuliah tidak terikat dengan semester, maka untuk menghapus kolom tersebut dapat digunakan *query* berikut:

```
ALTER TABLE matakuliah
DROP semester;
```

4. Menyimpan Data ke Dalam Tabel

Data-data berikut akan dimasukkan ke dalam tabel-tabel yang telah dibuat. Perintah atau *query* untuk menyimpan data-data tersebut ke dalam tabel yang telah ditentukan juga tertera di bawahnya.

Tabel: **mahasiswa**

Nim	nama_mahasiswa	jenis_ kelamin	tempat_lahir	tanggal_lahir	alamat	semester	jurusan
101	Alvin Drajat	L	Banyuwangi	1995-10-7	Jl. Widuri	6	S1 TI
102	Bella Paramitha	P	Bandung	1996-8-5	Jl. Bengawan	4	D3 MI
103	Rusdian Permana	L	Banyuwangi	1996-7-15	Jl. Solo	4	D3 MI
104	Citra Mandala	P	Banyuwangi	1997-1-13	Jl. Widuri	2	S1 TI

```
INSERT INTO mahasiswa
VALUES
('101','Alvin Drajat','L','Banyuwangi','1995-10-7','Jl. Widuri',6,'S1 TI'),
('102','Bella Paramitha','P','Bandung','1996-8-5','Jl. Bengawan',4,'D3 MI'),
('103','Rusdian Permana','L','Banyuwangi','1996-7-15','Jl. Solo',4,'D3 MI'),
('104','Citra Mandala','P','Banyuwangi','1997-1-13','Jl. Widuri',2,'S1 TI');
```

Tabel: **dosen**

id_dosen	nama_dosen	jenis_kelamin	pendidikan_terakhir
901	Bambang Wahyu Pribadi	L	S2
902	Deri Ahmad Muttaqien	L	S1
903	Firda Resta Maulina	P	S1
904	Heru Mulyadi	L	S3

```
INSERT INTO dosen
VALUES
('901','Bambang Wahyu Pribadi','L','S2'),
('902','Deri Ahmad Muttaqien','L','S1'),
('903','Firda Resta Maulina','P','S1'),
('904','Heru Mulyadi','L','S3');
```

Tabel: **matakuliah**

kode_matakuliah	nama_matakuliah	sks
001	Basis Data	4
002	Software Modelling	2
003	Web Programming	3
004	Agama	2

```
INSERT INTO matakuliah
VALUES
('001','Basis Data',4),
('002','Software Modelling',2),
('003','Web Programming',3),
('004','Agama',2);
```

Tabel: **kelas**

kode_matakuliah	id_dosen
001	901
001	901
003	901
002	904
002	904
003	903
004	

```
INSERT INTO kelas(kode_matakuliah,id_dosen)
VALUES
('001','901'),
('001','901'),
('003','901'),
('002','904'),
('002','904'),
('003','903');
```

```
INSERT INTO kelas(kode_matakuliah)
VALUES('004');
```

Tabel: **peserta_kuliah**

kode_kelas	nim
1	104
3	102
3	103
2	101
2	102
2	103

```
INSERT INTO peserta_kuliah(kode_kelas,nim)
VALUES
(1,'104'),
(3,'102'),
(3,'103'),
(2,'101'),
(2,'102'),
(2,'103');
```

Cobalah untuk menambahkan kembali data baru dengan kode kelas = **1** dan NIM = **104** ke dalam tabel **peserta_kuliah**! Apakah data tersebut tetap dapat tersimpan?

Perhatikan data yang ada pada kolom kode_kelas di tabel kelas dan kode_peserta di tabel peserta_kuliah! Apakah data yang muncul secara otomatis tersebut berupa data yang berurutan?

5. Menampilkan Data di Dalam Tabel

Untuk menampilkan data seluruh mahasiswa dapat dilakukan dengan menggunakan *query* berikut:

```
SELECT * FROM mahasiswa;
```

Namun jika yang dibutuhkan hanyalah data **nim** dan **nama** mahasiswa saja, maka *query* yang digunakan adalah sebagai berikut:

```
SELECT nim, nama FROM mahasiswa;
```

Perlu dicatat bahwa penggunaan baris baru (enter) pada *query* SQL tidak akan mempengaruhi hasil. Dengan kata lain *query* SQL yang ditulis dalam satu baris utuh dengan yang diatur menggunakan garis baru dianggap sama. Hanya saja penyajian *query* yang diatur dengan garis baru terlihat lebih rapi dan mudah dibaca.

6. Menampilkan / Mencari Data di Dalam Tabel Menggunakan Kriteria Tertentu

Apabila ingin menampilkan data **kelas** yang diampu oleh dosen dengan ID **901**, maka *query* yang digunakan adalah sebagai berikut:

```
SELECT * FROM kelas
WHERE id_dosen = '901';
```

Apabila ingin mengetahui daftar **nama matakuliah** yang **sks**-nya lebih dari **2**, maka *query* yang digunakan adalah sebagai berikut:

```
SELECT nama_matakuliah FROM matakuliah
WHERE sks > 2;
```

Untuk mengetahui **kelas** yang belum ditentukan **dosen** pengampunya maka dapat dilakukan dengan menggunakan *query* berikut:

```
SELECT * FROM kelas
WHERE id_dosen IS NULL;
```

Untuk mengetahui data **dosen pria** dengan pendidikan terakhir **S1**, maka dapat dilakukan dengan menggunakan *query* berikut:

```
SELECT * FROM dosen
WHERE jenis_kelamin = 'L' AND pendidikan_terahir = 'S1';
```

7. Mengeliminasi Data Kembar dari Hasil Pencarian Data

Untuk mengetahui ID dosen yang telah ditugaskan untuk mengampu suatu kelas, maka dapat dilakukan dengan menggunakan *query* berikut:

```
SELECT DISTINCT id_dosen FROM kelas
WHERE id_dosen IS NOT NULL;
```

Sedangkan untuk mengetahui data matakuliah apa saja yang diampu oleh setiap dosen dapat dilakukan dengan menggunakan *query* berikut:

```
SELECT DISTINCT kode_matakuliah, id_dosen
FROM kelas
WHERE id_dosen IS NOT NULL;
```

8. Mengurutkan Data

Contoh pengurutan pada hasil pencarian data di dalam basis data adalah pengurutan data mahasiswa secara *descending* berdasarkan namanya. *Query* yang digunakan untuk kasus tersebut adalah sebagai berikut:

```
SELECT * FROM mahasiswa
ORDER BY nama_mahasiswa DESC;
```

Contoh lainnya adalah pengurutan data matakuliah berdasarkan besaran SKS-nya secara *ascending*. Untuk kasus tersebut bentuk *query*-nya adalah sebagai berikut:

```
SELECT * FROM matakuliah
ORDER BY sks;
```

9. Mengubah Data

Sebagai contoh, apabila terjadi perubahan terhadap data dosen dengan ID **902** dimana pendidikan terakhirnya diperbaharui menjadi **S2**, maka *query* yang dapat digunakan untuk kasus tersebut adalah sebagai berikut:

```
UPDATE dosen
SET pendidikan_terakhir = 'S2'
WHERE id_dosen = '902';
```

10. Menghapus Data

Apabila mahasiswa dengan NIM **102** tidak jadi mengikuti kelas dengan kode **3**, maka *query* yang digunakan untuk menghapus data tersebut adalah sebagai berikut:

```
DELETE FROM peserta_kuliah
WHERE nim = '102' AND kode_kelas = 3;
```

11. Menghapus Tabel

Apabila tabel **kelas** ingin dihapus dari dalam *database*, maka *query* yang digunakan adalah sebagai berikut:

```
DROP TABLE kelas;
```

12. Menghapus Basis Data

Apabila *database* **kampus** sudah tidak terpakai lagi dan ingin dihapus dari dalam mesin atau komputer, maka *query* yang digunakan adalah sebagai berikut:

```
DROP DATABASE kampus;
```

E. TUGAS

Tuliskan *query* untuk menyelesaikan kasus-kasus berikut di dalam laporan praktikum!

1. Buat sebuah *database* dengan nama **onlineshop**!
2. Buat tabel-tabel berikut di dalam *database* **onlineshop**:

Tabel: **penjual**

Kolom	Tipe Data	Panjang	Spesifikasi
id_penjual	CHAR	5	Primary key
nama	VARCHAR	30	Not null
alamat	TEXT	-	Not null
no_telp	CHAR	12	-
email	VARCHAR	100	-
bank	VARCHAR	10	Not null
no_rekening	VARCHAR	20	Not null

Tabel: **pembeli**

Kolom	Tipe Data	Panjang	Spesifikasi
id_pembeli	CHAR	5	Primary key
nama	VARCHAR	30	Not null
alamat	TEXT	-	Not null
no_telp	CHAR	12	-
email	VARCHAR	100	-
bank	VARCHAR	10	Not null
no_rekening	VARCHAR	20	Not null

Tabel: **produk**

Kolom	Tipe Data	Panjang	Spesifikasi
id_produk	INT	-	Primary key, auto increment
nama	VARCHAR	70	Not null
kode_kategori	INT	-	Not null
deskripsi	TEXT	-	-
harga	INT	-	Default = 1000
berat	INT	-	Default = 100
kondisi	ENUM	Baru, Bekas	Not null

Tabel: **kategori_produk**

Kolom	Tipe Data	Panjang	Spesifikasi
kode_kategori	INT	-	Primary key, auto increment
kategori	VARCHAR	30	Not null

Tabel: **transaksi**

Kolom	Tipe Data	Panjang	Spesifikasi
id_transaksi	INT	-	Primary key, auto increment
id_pembeli	CHAR	5	Not null
id_produk	INT	-	Not null
jumlah	INT	3	Default = 1
total_bayar	INT	-	Not null
waktu_transaksi	TIMESTAMP	-	-

Tabel: **daftar_status_transaksi**

Kolom	Tipe Data	Panjang	Spesifikasi
kode_status	CHAR	3	Primary key
status	VARCHAR	30	Not null

Tabel: **progres_transaksi**

Kolom	Tipe Data	Panjang	Spesifikasi
id_transaksi	INT	-	Not null
kode_status	CHAR	3	Not null
waktu	TIMESTAMP	-	-

3. Tambahkan kolom id_penjual pada tabel produk agar dapat diketahui siapa yang memiliki produk tersebut! Sesuaikan tipe data dan spesifikasi kolom tersebut seperti pada tabel penjual.
4. Ubah jumlah maksimum karakter pada nomor rekening pembeli maupun penjual menjadi 15 karakter!
5. Kolom waktu_transaksi pada tabel transaksi tidak lagi dibutuhkan karena waktu transaksi dapat dilihat dari waktu progres transaksi tersebut ketika masih dalam kondisi *check out*. Hapus kolom waktu_transaksi tersebut!
6. Masukkan data-data berikut ke dalam tabel-tabel yang telah dibuat sebelumnya:

Tabel: **penjual**

id_penjual	nama	alamat	no_telp	email	bank	no_rekening
J0001	Eko Shop	Jogjakarta	081134567654	ekoshop@gmail.com	BNI	067543
J0002	Max Comp	Surabaya	0227654321	max.comp@ymail.com	BRI	0111423
J0003	Juni Kids	Semarang	086685410092	juni.kids@gmail.com	BCA	000652431

Tabel: **pembeli**

id_pembeli	nama	alamat	no_telp	email	bank	no_rekening
B0001	Sulistina	Banyuwangi	083421657888		BCA	000976272
B0002	Hariyono	Semarang	085436379812	hari@ymail.com	BCA	000765321
B0003	Budi Wawo	Kupang	082187453096	budi.ww@gmail.com	BRI	0553421
B0004	Miftah Sani	Malang	088463521788	misan@gmail.com	BNI	076351

Tabel: **kategori_produk**

kategori
Fashion, Pakaian, & Aksesoris
Kecantikan & Kesehatan
Dapur & Rumah Tangga
Perawatan Bayi

kategori
Handphone & Tablet
Laptop/Komputer & Aksesoris
Elektronik
Kamera, Foto, & Video
Otomotif & Olahraga
Office & Stationery
Souvenir, Kado, & Hadiah
Mainan & Hobi
Makanan & Minuman
Buku
Software
Film, Musik, & Game
Produk Lainnya

Tabel: **produk**

nama	kode_kategori	deskripsi	harga	berat	kondisi	id_penjual
Mouse	6	Mouse untuk gamer	150000	100	baru	J0002
HDD External	6	HDD 1 TB mantap	785900	130	baru	J0002
Laptop ASUS	6	Laptop ASUS K42JE pemakaian 1 tahun	3500000	1000	bekas	J0002
Teddy bear	12	Boneka teddy bear warna coklat bersih cocok untuk teman si kecil	450000	200	baru	J0003

Tabel: **daftar_status_transaksi**

kode_status	status
T01	Check out
T02	Konfirmasi pembayaran
T03	Pembayaran terverifikasi
T04	Pesanan sedang diproses
T05	Pesanan dikirim
T06	Pesanan tiba di tujuan
T07	Transaksi selesai

Tabel: **transaksi**

id_pembeli	id_produk	jumlah	total_bayar
B0002	2	1	795900
B0002	1	2	310000
B0004	4	1	465000

7. Masukkan data-data berikut ke tabel **progres_transaksi** satu per satu kemudian tampilkan seluruh data pada tabel tersebut! Gambarkan tabel serta isi data yang ditampilkan! Perhatikan nilai yang muncul pada kolom waktu!

id_transaksi	kode_status
1	T01
2	T01
1	T02
2	T02
1	T03

id_transaksi	kode_status
2	T02
3	T01
3	T02
3	T03
3	T04
3	T05

8. Tampilkan daftar nama bank yang digunakan oleh pembeli!
9. Tampilkan nama pembeli yang belum mengisikan alamat emailnya!
10. Tampilkan data transaksi dimana jumlah pembelian produknya lebih dari 1 produk!
11. Tampillkan daftar nama produk beserta harga satuannya! Urutkan dari produk yang paling mahal ke produk yang paling murah!
12. Perbaharui data harga laptop ASUS yang dijual oleh penjual dengan ID J0002 menjadi Rp3.000.000!
13. Hapus data pembeli dengan kode B0003!
14. Hapus tabel penjual dari dalam *database* onlineshop!
15. Hapus *database* onlineshop!