

## | Apa itu Basis

Basis adalah dasar atau fondasi dari suatu proses atau argumen.

## | Apa itu Data

Sekumpulan informasi yang diperoleh dari pengamatan, yang dapat berupa simbol, angka, dan properti.

## | Apa itu Basis Data

Basis data adalah kumpulan data yang disusun secara sistematis dan terorganisir sehingga mudah untuk diakses, dikelola, dan diperbarui. Basis data biasanya dikelola menggunakan sistem manajemen basis data.

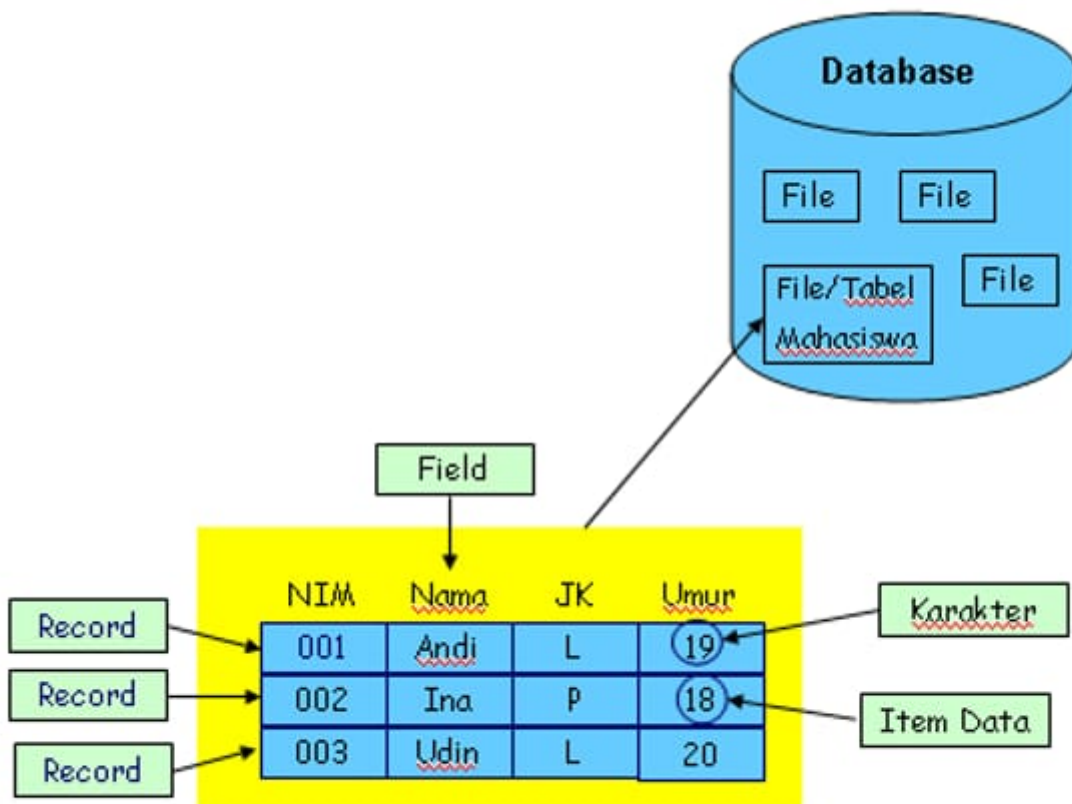
## | Peranan Sekolah

Basis data sekolah adalah kumpulan informasi tentang sekolah, seperti lokasinya, informasi kontak, nomor pendaftaran, dan jenis program yang ditawarkan. Informasi ini dapat berguna untuk berbagai tujuan, seperti:

- Membantu orang tua dan siswa menemukan sekolah di daerah mereka yang memenuhi kebutuhan dan preferensi mereka
- Memfasilitasi penelitian tentang pendidikan dan kinerja sekolah dengan menyediakan data pendaftaran, nilai ujian, tingkat kelulusan, dll.
- Mendukung pekerjaan lembaga dan organisasi pemerintah yang mendanai atau mengatur sekolah dengan menyediakan data mengenai demografi dan kinerja sekolah
- Memungkinkan pendidik dan administrator untuk berbagi informasi dan sumber daya dengan sekolah lain, seperti praktik terbaik, materi kurikulum, dan peluang pengembangan profesional.

## | Data base

## | Contoh tabel



## Struktur

Struktur Basis Data merupakan serangkaian pengetahuan tentang komponen penyusun data beserta hubungan komponen tersebut. Struktur basis data juga dapat didefinisikan sekumpulan cara/peralatan/tool untuk mendeskripsikan data-data, hubungannya satu sama lain, serta batas konsistensi. Struktur basis data digunakan agar pemrosesan data menjadi lebih efisien. Struktur tersebut meliputi file, table, field, record indeks.

Nama	JK	Sekolah
Jeno	L	SMKN 7 Makassar
Jaemin	L	SMKN 7 Makassar
Jisung	L	SMKN 7 Makassar

## Menggunakan XAMPP

1. Buka aplikasi XAMPP
2. Klik `<star>` di MYSQL
3. Klik `<Shell>`

4. Masuk kedatabase dengan akun administrator `<mysql -u root -p>`  
password kosong jadi silahkan langsung enter
5. Buat Database
  - Create Database
2. Tampilkan Database
  - `<show database>`
3. Hentikan Database
  - `<drop database [nama_database]>`
4. Menggunakan Database
  - `<use [nama_database]>`

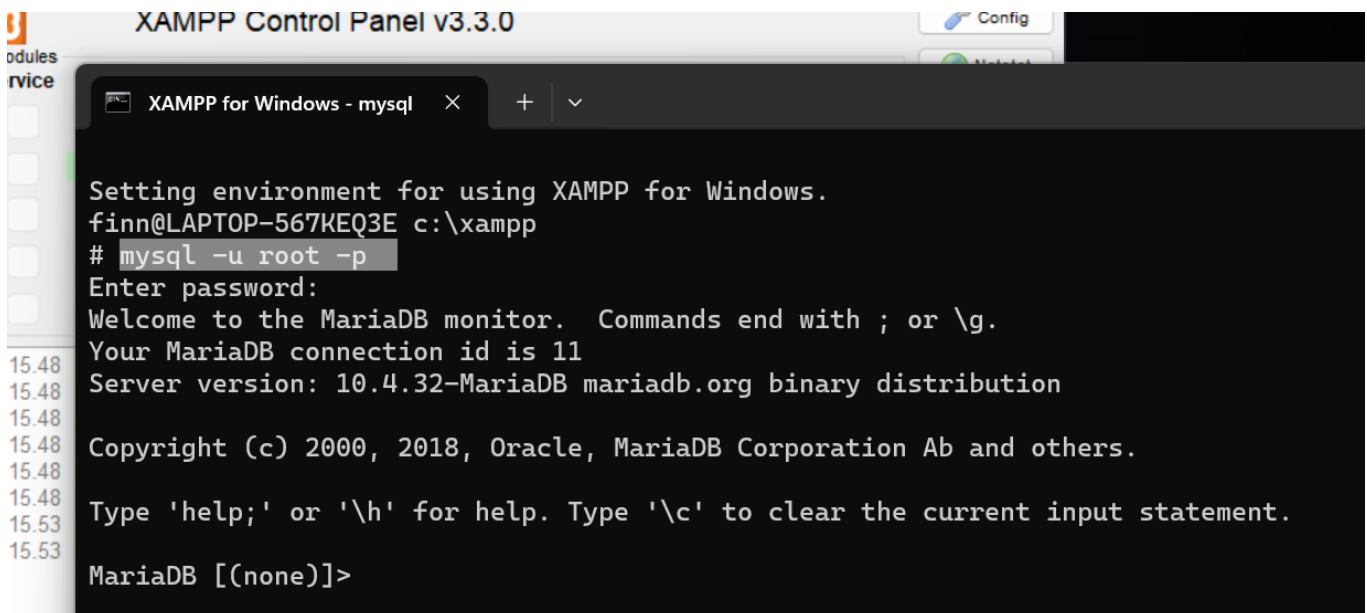
## Referensi video youtube

<https://youtu.be/3UN1ZNvqEt8?si=tAbr0KJ1D1S3iKwG>

## Penggunaan Awal MySQL

- Query  
`<mysql -u root -p>`

## Hasil



```
XAMPP Control Panel v3.3.0
modules
service

XAMPP for Windows - mysql
Setting environment for using XAMPP for Windows.
finn@LAPTOP-567KEQ3E c:\xampp
# mysql -u root -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.
MariaDB [(none)]>
```

## Analisis

- `<mysql>` Salah satu aplikasi database server dengan bahasa pemrograman structured query language (`SQL`) yang berfungsi untuk mengelola data secara terstruktur dan sistematis.
- `<-u root>` Bagian ini mengeset pengguna (user) yang akan digunakan saat terhubung ke server MYSQL. Dalam contoh ini, pengguna yang digunakan adalah `"root"`. Pengguna `"root"` biasanya memiliki hak akses penuh ke server MYSQL dan dapat melakukan tindakan administratif.
- `<-p>` Opsi ini digunakan untuk meminta kata sandi (password) setelah perintah dijalankan ini adalah langkah keamanan yang umum digunakan untuk memastikan hanya pengguna yang sah yang dapat mengakses server MYSQL. Setelah kita menekan Enter setelah perintah ini, kita akan diminta memasukkan kata sandi untuk pengguna `"root"`.

## Kesimpulan

Perintah `mysql -u root -p` digunakan untuk masuk ke sistem manajemen basis data MySQL dengan mengidentifikasi pengguna (user) 'root' dan meminta kata sandi (password) secara interaktif.

## DataBase

Database (basis data) adalah kumpulan terstruktur dari informasi yang disimpan secara elektronik dalam sistem komputer. Database dirancang untuk menyimpan, mengatur, dan mengelola data dengan cara yang efisien dan dapat diakses.

## Buat DataBase

Database adalah sekumpulan data yang dikelola berdasarkan ketentuan tertentu yang saling berkaitan sehingga memudahkan dalam pengelolannya.

## Struktur Query

```
create databases [nama_tabel];
```

## Contoh Query

```
create databases XI_RPL_1;
```

## Hasil

```
MariaDB [(none)]> create database xi_rpl_1;
Query OK, 1 row affected (0.002 sec)
```

## Analisis

- `CREATE DATABASE` adalah perintah untuk membuat database baru.
- `XI_RPL_1` adalah nama yang Anda pilih untuk database baru Anda. Tanda kurung siku `<("[ ]")>` digunakan di sini untuk menghindari kesalahan jika nama database mengandung karakter spesial atau spasi. Namun, perlu dicatat bahwa tidak semua DBMS mengizinkan penggunaan tanda kurung siku dalam nama database, jadi pastikan untuk menyesuaikan sintaks dengan DBMS yang Anda gunakan.

## | Kesimpulan

Kesimpulan dari perintah `"CREATE DATABASE XI_RPL_1;"` adalah bahwa perintah tersebut digunakan untuk membuat sebuah database baru dengan nama `"XI_RPL_1"`.

## | Tampilkan DataBase

## | Struktur Query

```
show [nama_databases];
```

## | Contoh Query

```
show databases;
```

## | Hasil

```
MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| phpmyadmin |
| test |
| xi_rpl_1 |
+-----+
6 rows in set (0.058 sec)
```

## | Analisis

`SHOW DATABASE` digunakan untuk menampilkan daftar database yang ada dalam sistem manajemen basis data (DBMS). Perintah ini dapat digunakan di beberapa DBMS seperti MySQL, PostgreSQL, dan beberapa DBMS lainnya. Namun, perintahnya dapat sedikit berbeda tergantung.

## | Kesimpulan

Kesimpulan dari perintah `"SHOW DATABASES;"` adalah bahwa perintah tersebut digunakan untuk menampilkan daftar database yang ada dalam sistem manajemen basis data yang sedang digunakan. Perintah ini akan menghasilkan output berupa daftar nama-nama database yang tersedia.

## | Hapus DataBase

### | Struktur Query

```
drop database [nama_dataasbe]
```

### | Contoh Query

```
drop database xi_rpl_1;
```

### | Hasil :

```
MariaDB [(none)]> drop database xi_rpl_1;  
Query OK, 0 rows affected (0.002 sec)
```

## | Analisis

`<DROP DATABASE [nama_database]>` digunakan dalam sistem manajemen basis data (DBMS) untuk menghapus sebuah database beserta semua objek yang terkait dengan database tersebut, seperti tabel, indeks, tampilan, prosedur tersimpan, dan lain-lain.

## | Kesimpulan

Kesimpulan dari perintah `"DROP DATABASE XI_RPL_1;"` adalah bahwa perintah tersebut digunakan untuk menghapus database dengan nama `"XI_RPL_1"` dari sistem manajemen basis data yang sedang digunakan.

## | Gunakan DataBase

### | Struktur Query

```
use [nama_database]
```

## | Contoh Query

```
use xi_rpl_1;
```

## | Hasil :

```
MariaDB [(none)]> use xi_rpl_1;  
Database changed  
MariaDB [xi_rpl_1]>
```

## | Analisis

`USE [nama_database]` digunakan dalam sistem manajemen basis data (DBMS) untuk beralih atau memilih database yang akan digunakan. Ketika Anda menggunakan perintah `<USE>` diikuti dengan nama database, DBMS akan mengarahkan semua perintah dan operasi selanjutnya pada database yang ditentukan.

## | Kesimpulan

Kesimpulan dari perintah `"USE XI_RPL_1;"` adalah bahwa perintah tersebut digunakan untuk memilih atau beralih ke database dengan nama `"XI_RPL_1"` dalam sistem manajemen basis data yang sedang digunakan.

## | Tipe Data

### | Angka

- Tipe data ini digunakan untuk merepresentasikan bilangan bulat tanpa bagian desimal.
- Contoh: 0, 42, -10

### | Teks

- Tipe data ini digunakan untuk merepresentasikan bilangan dengan bagian desimal.
- Contoh: 3.14, 2.5, -0.5

### | Tanggal

- `Date` digunakan untuk menyimpan informasi tentang tanggal, biasanya terdiri dari hari, bulan, dan tahun seperti 30 Januari 2024

- `Time` digunakan untuk menyimpan informasi tentang waktu dalam sehari, biasanya terdiri dari jam, menit, detik, dan milidetik seperti 14:30:45.500
- `DateTime` menggabungkan informasi tanggal dan waktu dalam satu objek, biasanya terdiri dari hari, bulan, tahun, jam, menit, detik, dan milidetik seperti 30 Januari 2024 14:30:45.500

## | Boolean

- Tipe data ini hanya memiliki dua nilai yang mungkin, yaitu `true` (benar) dan `false` (salah).

## | Tabel

## | Buat Tabel

## | Struktur Query

```
create table [nama table](
    namakolom_1 tipedata(lebar) cons,
    namakolom_2 tipedata(lebar) cons,
    namakolom_3 tipedata(lebar) cons,
)
```

## | Contoh Query

```
nama_mobil varchar(15) primary key not null,
plat_mobil char(18) not null unique,
warna_mobil varchar(18) not null unique);
```

## | Hasil

```
MariaDB [rental_mada]> describe mobil;
```

Field	Type	Null	Key	Default	Extra
nama_mobil	varchar(15)	NO	PRI	NULL	
plat_mobil	char(10)	NO	UNI	NULL	
warna_mobil	varchar(10)	NO	UNI	NULL	

3 rows in set (0.038 sec)

## | Analisis

1. `nama_mobil varchar(15) primary key not null`: Ini adalah kolom dengan nama "nama\_mobil" yang memiliki tipe data VARCHAR dengan panjang maksimum 15 karakter. Primary key menandakan bahwa kolom ini akan berfungsi sebagai kunci utama untuk



mengidentifikasi setiap baris dalam tabel. "Not null" menunjukkan bahwa kolom tidak boleh memiliki nilai null, yang berarti setiap baris harus memiliki nilai yang valid untuk kolom ini.

2. `plat_mobil char(18) not null unique`: Ada beberapa kesalahan ketik dalam definisi kolom ini. Saya berasumsi bahwa maksud Anda adalah `plat_mobil char(18) not null unique`. Kolom ini memiliki nama "plat\_mobil" dan menggunakan tipe data CHAR dengan panjang tetap 18 karakter. "Not null" menunjukkan bahwa kolom ini tidak boleh memiliki nilai null. "Unique" menandakan bahwa setiap nilai dalam kolom harus unik, artinya tidak ada dua baris dalam tabel yang memiliki nilai yang sama untuk kolom ini.
3. `warna_mobil varchar(18) not null unique`: Ini adalah kolom dengan nama "warna\_mobil" yang menggunakan tipe data VARCHAR dengan panjang maksimum 18 karakter. "Not null" menunjukkan bahwa kolom ini tidak boleh memiliki nilai null. "Unique" menandakan bahwa setiap nilai dalam kolom harus unik, sehingga tidak ada dua baris dalam tabel yang memiliki nilai yang sama untuk kolom ini.

## | Kesimpulan

1. `nama_mobil varchar(15) primary key not null`,
  - Tipe data: VARCHAR(15), Atribut: Primary key (utama), tidak dapat bernilai NULL (not null). Kolom ini digunakan sebagai kunci utama (primary key) untuk tabel dan harus diisi dengan nilai yang unik. Panjang karakter maksimum adalah 15.
2. `plat_mobil char(18) not null unique`
  - Tipe data: CHAR(18), Atribut: Tidak dapat bernilai NULL (not null), harus memiliki nilai unik (unique). Kolom ini harus diisi dan memuat nilai unik dengan panjang karakter tetap sebanyak 18 karakter.
3. `warna_mobil varchar(18) not null unique`,
  - Tipe data: VARCHAR(18), Atribut: Tidak dapat bernilai NULL (not null), harus memiliki nilai unik (unique). Kesimpulan: Kolom ini harus diisi dan memuat nilai unik dengan panjang karakter maksimum sebanyak 18.

## | Tampilkan Struktur Tabel

## | Struktur Query

```
desc [nama_table];
```

## | Contoh Query

```
desc pelanggan;
```

## | Hasil

```
MariaDB [rental_fina]> desc pelanggan;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_pelanggan  | int(4)        | NO   | PRI | NULL    |       |
| nama_depan    | varchar(25)   | NO   |     | NULL    |       |
| nama_belakang | varchar(25)   | NO   |     | NULL    |       |
| no_telp       | char(12)      | YES  | UNI | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.037 sec)
```

## | Analisis

`desc pelanggan;` : dapat melihat secara detail karakteristik dari setiap kolom dalam tabel tersebut, termasuk nama kolom, tipe data, panjang maksimum (jika berlaku), dan konstrain khusus seperti PRIMARY KEY, UNIQUE, atau NOT NULL yang diterapkan pada setiap kolom.

## | Kesimpulan

perintah tersebut memberikan gambaran tentang bagaimana tabel "Pelanggan" telah didefinisikan dalam basis data.

## | Menampilkan Tabel

## | Struktur Query

```
show tables;
```

## | Contoh Query

```
show tables;
```

## | Hasil

```
MariaDB [rental_fina]> show tables;
+-----+
| Tables_in_rental_fina |
+-----+
| pelanggan              |
+-----+
1 row in set (0.028 sec)
```

## | Analisis

`show tables;` : untuk menampilkan semua tabel yang ada dalam database yang sedang aktif.

## I Kesimpulan

Perintah "`SHOW TABLES;`" akan menghasilkan output berupa daftar nama-nama tabel yang tersedia dalam database yang sedang digunakan. Output ini memberikan informasi tentang tabel-tabel yang ada dalam database dan memungkinkan pengguna untuk melihat struktur dan konten data dalam tabel-tabel tersebut.

## I QnA

### ? Mengapa hanya kolom `id_pelanggan` yang menggunakan constraint **PRIMARY KEY**? >

kolom `id_pelanggan` menggunakan constraint **PRIMARY KEY** karena dianggap sebagai atribut yang unik dan mengidentifikasi secara unik setiap baris dalam tabel pelanggan. Constraint **PRIMARY KEY** digunakan untuk memastikan bahwa tidak ada nilai yang duplikat atau **NULL** dalam kolom tersebut.

### ? Mengapa pada kolom `no_telp` yang menggunakan tipe data `char` bukan `varchar`? >

tipe data `char` atau `varchar` untuk kolom `no_telp` tergantung pada kebutuhan dan karakteristik data yang akan disimpan dalam kolom tersebut.

### ? Mengapa hanya kolom `no_telp` yang menggunakan constraint **UNIQUE**? >

Kolom `no_telp` menggunakan constraint **UNIQUE** karena dianggap harus memiliki nilai yang unik di setiap baris dalam tabel. dalam menggunakan constraint **UNIQUE** pada kolom `no_telp` memastikan bahwa tidak ada dua pelanggan dengan nomor telepon yang sama.

### ? Mengapa kolom `no_telp` tidak memakai constraint **NOT NULL**, sementara kolom lainnya menggunakan constraint tersebut? >

pada kolom seperti `no_telp`, ada situasi di mana nomor telepon mungkin tidak tersedia atau tidak diketahui untuk beberapa pelanggan. Dalam kasus ini, memaksa kolom `no_telp` menggunakan constraint **NOT NULL** akan menjadi tidak praktis dan dapat membatasi fleksibilitas data.

### ? Perbedaan **PK & UNIQUE**? >

PK digunakan untuk mengidentifikasi secara unik setiap baris dalam tabel. Setiap tabel biasanya memiliki satu PRIMARY KEY yang berfungsi sebagai pengenal utama untuk baris-baris tersebut.

UNIQUE digunakan untuk memastikan bahwa setiap nilai dalam kolom tertentu adalah unik, tetapi tidak digunakan untuk mengidentifikasi secara unik setiap baris dalam tabel.

## | Insert

### | Insert 1 data

### | Struktur

```
INSERT INTO [nama_tabel]
VALUES (nilai1, nilai2, nilai3, ...);
```

## | Contoh

```
INSERT INTO pelanggan
values (1, "mada", "jeno", "082195305736");
```

## | Hasil

```
MariaDB [rental_fina]> insert into pelanggan
-> values (1, "mada", "jeno", '082195305736');
Query OK, 1 row affected (0.153 sec)
```

## | Analisis

1. Kolom pertama: mungkin merupakan kolom ID pelanggan dengan tipe data numerik.
2. Kolom kedua: mungkin merupakan kolom nama dengan tipe data teks.
3. Kolom ketiga: mungkin merupakan kolom nama belakang dengan tipe data teks.
4. Kolom keempat: mungkin merupakan kolom nomor telepon dengan tipe data teks.

`INSERT INTO` akan menyisipkan nilai-nilai yang Anda berikan ke dalam tabel "pelanggan" sesuai dengan urutan kolom yang ada. Pastikan bahwa struktur tabel "pelanggan" sudah sesuai dengan urutan dan tipe data yang digunakan dalam perintah tersebut.

## | Kesimpulan

`INSERT INTO pelanggan VALUES (1, "mada", "jeno", "082195305736");`, dapat disimpulkan bahwa Anda sedang mencoba menyisipkan satu baris data baru ke dalam tabel "pelanggan". Jika struktur tabel "pelanggan" benar dan sesuai dengan perintah tersebut, maka data baru dengan nilai-nilai yang Anda berikan akan dimasukkan ke dalam tabel "pelanggan".

## | Insert > 1 data

## | Struktur

```
Insert into [nama_table]
Values (nilai1, nilai2, nilai3, nilai4)
       (nilai1, nilai2, nilai3, nilai4)
       (nilai1, nilai2, nilai3, nilai4)
```

## | Contoh

```
insert into pelanggan
values (5, "mada", "jeno", '0835515796087'),
(4, "rafia", "nolan", '084255669897'),
(3, "gusion", "yusin", '08646903215');
```

## | Hasil

```
MariaDB [rental_fina]> insert into pelanggan
-> values (5, "mada", "jeno", '0835515796087'),
-> (4, "rafia", "nolan", '084255669897'),
-> (3, "gusion", "yusin", '08646903215');
Query OK, 3 rows affected, 1 warning (0.044 sec)
Records: 3  Duplicates: 0  Warnings: 1
```

## | Analisis

- `INSERT INTO pelanggan` : Menentukan bahwa Anda ingin memasukkan data ke dalam tabel "pelanggan".
- `VALUES` : Menunjukkan bahwa Anda memberikan nilai untuk dimasukkan ke dalam kolom yang ditentukan.
- Nilai dalam tanda kurung mewakili masing-masing baris yang akan dimasukkan ke dalam tabel "pelanggan".
- Baris pertama memiliki nilai `5, "mada", "jeno", '0835515796087'`.
- Baris kedua memiliki nilai `4, "rafia", "nolan", '084255669897'`.
- Baris ketiga memiliki nilai `3, "gusion", "yusin", '08646903215'`.

## Kesimpulan

Baris pertama memiliki nilai ID 5, nama depan "mada", nama belakang "jeno", dan nomor telepon '0835515796087'. Baris kedua memiliki nilai ID 4, nama depan "rafia", nama belakang "nolan", dan nomor telepon '084255669897'. Baris ketiga memiliki nilai ID 3, nama depan "gusion", nama belakang "yusin", dan nomor telepon '08646903215'.

## Select

## Seluruh Data

## Struktur

```
select * from [nama_tabel];
```

## Contoh

```
select * from pelanggan;
```

## Hasil

```
MariaDB [rental_fina]> select * from pelanggan;
```

id_pelanggan	nama_depan	nama_belakang	no_telp
1	mada	jeno	082195305736
3	gusion	yusin	08646903215
4	rafia	nolan	08425569897
5	mada	jeno	083551579608

```
4 rows in set (0.000 sec)
```

## Analisis

Kueri SQL "SELECT \* FROM pelanggan;" adalah query sederhana yang digunakan untuk mengambil semua kolom dan record dari tabel bernama "pelanggan" dalam database. Izinkan saya menguraikan pertanyaannya untuk Anda:

- **SELECT \*** : digunakan untuk memilih semua kolom dari tabel yang ditentukan.
- **FROM pelanggan** : Bagian ini menentukan tabel untuk mengambil data. Dalam tabelnya diberi nama "pelanggan".

## Kesimpulan

program akan mengambil dan menampilkan semua data yang tersimpan dalam tabel Pelanggan, termasuk setiap kolom dan setiap baris yang ada dalam tabel tersebut.

## | Data Kolom Tertentu

## | Struktur

```
select [nama_kolom1], [nama_kolom2], ..., [nama_kolom_n]
from [nama_tabel],
```

## | Contoh

```
select nama_depan from pelanggan;
```

## | Hasil

```
MariaDB [rental_fina]> select nama_depan from pelanggan;
+-----+
| nama_depan |
+-----+
| mada       |
| gusion     |
| rafia      |
| mada       |
+-----+
4 rows in set (0.000 sec)
```

## | Analisis

- `Select` merupakan query yang digunakan untuk menampilkan hasil `insert`
- `nama_depan` nama kolom dalam tabel database yang mungkin menyimpan informasi tentang nama depan dari pelanggan.
- `from` query yang digunakan untuk memberikan penanda bahwa table mana yang akan di tampilkan
- `pelanggan` merupakan nama table yang isi nya akan di tampilkan

## | Kesimpulan

Hasilnya akan berupa daftar `nama_depan` dari semua pelanggan yang terdaftar dalam tabel tersebut.

## | Klausula WHERE

## Struktur

```
select [nama_kolom] from [nama_tabel]  
where [kondisi];
```

## Contoh

```
select nama_depan from pelanggan  
where id =2;
```

## Hasil

```
MariaDB [rental_fina]> select nama_depan from pelanggan  
-> where id_pelanggan=1;  
+-----+  
| nama_depan |  
+-----+  
| mada      |  
+-----+  
1 row in set (0.041 sec)
```

## Analisis

- **Select** merupakan query yang digunakan untuk menampilkan hasil **insert**
  - **id\_pelanggan, nama\_depan** nama kolom dalam tabel database yang mungkin menyimpan informasi tentang nama depan dari pelanggan.
  - **from** query yang digunakan untuk memberikan penanda bahwa table mana yang akan di tampilkan
  - **pelanggan** merupakan nama table yang isi nya akan di tampilkan
  - **where** untuk menyaring baris data berdasarkan kondisi tertentu.
  - **id\_pelanggan=1** hanya baris-baris data di mana nilai kolom

## Kesimpulan

hasilnya akan berisi ID dan nama depan pelanggan yang memiliki ID tertentu

## Update

## Struktur

```
UPDATE nama_tabel SET nama_kolom WHERE kondisi;
```



## Contoh

```
UPDATE pelanggan SET no_telp="085358639358" WHERE id_pelanggan="1";
```

## Hasil

```
MariaDB [rental_fina]> select * from pelanggan;
```

id_pelanggan	nama_depan	nama_belakang	no_telp
1	mada	jeno	085358639358
2	mada	jeno	083057025739
3	gusion	yusin	08646903215
4	rafia	nolan	08425569897
5	mada	jeno	083551579608

5 rows in set (0.001 sec)

## Analisis

- `UPDATE pelanggan` : Ini adalah klausa yang menentukan tabel mana yang akan diperbarui. Dalam kasus ini, tabel yang diperbarui adalah "pelanggan".
- `SET no_telp="085358639358"` : Ini adalah klausa yang menentukan kolom mana yang akan diperbarui dan nilai baru yang akan diberikan. Dalam hal ini, kolom yang diperbarui adalah "no\_telp" dan nilainya diubah menjadi "085358639358".
- `WHERE id_pelanggan="1"` : Ini adalah klausa opsional yang digunakan untuk membatasi baris mana yang akan diperbarui. Dalam hal ini, perubahan hanya akan diterapkan pada baris dengan nilai "id\_pelanggan" yang sama dengan "1".

## Kesimpulan

nomor telepon (no\_telp) dari pelanggan dengan ID "1" akan diubah menjadi "085358639358". Perintah tersebut mengupdate data pada tabel "pelanggan" dan mengaplikasikan perubahan hanya pada baris dengan nilai "id\_pelanggan" yang sama dengan "1".

## Delete

## Struktur

```
DELETE FROM nama_tabel WHERE kondisi;
```

## Contoh

```
DELETE FROM pelanggan WHERE id_pelanggan="2";
```

## Hasil

```
MariaDB [rental_fina]> select * from pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
|          1 | mada      | jeno          | 085358639358 |
|          3 | gusion    | yusin         | 08646903215  |
|          4 | rafia     | nolan         | 08425569897  |
|          5 | mada      | jeno          | 083551579608 |
+-----+-----+-----+-----+
4 rows in set (0.001 sec)
```

## Analisis

- `DELETE FROM pelanggan` : Ini adalah klausa yang menentukan tabel mana yang akan dihapus datanya. Dalam kasus ini, data akan dihapus dari tabel "pelanggan".
- `WHERE id_pelanggan="2"` : Ini adalah klausa opsional yang digunakan untuk membatasi baris mana yang akan dihapus. Dalam hal ini, baris dengan nilai "id\_pelanggan" yang sama dengan "2" akan dihapus.

## Kesimpulan

jika ingin menghapus baris table kalian bisa menggunakan query `delete` dengan struktur yaitu `delete from nama_table where kondisi;`

## Hapus Tabel

### Struktur Query

```
drop table [nama_tabel]
```

## Contoh Query

```
drop table mobil;
```

## Hasil

```
MariaDB [rental_fina]> drop table mobil;
Query OK, 0 rows affected (0.095 sec)
```

## | Analisis

- `"DROP TABLE"` : Ini adalah perintah SQL yang digunakan untuk menghapus sebuah tabel dari basis data.
- `"mobil"` : Ini adalah nama tabel yang ingin Anda hapus. Dalam kasus ini, tabel yang bernama "mobil" akan dihapus.

## | Kesimpulan

`"DROP TABLE"` adalah perintah yang digunakan untuk menghapus tabel dari sebuah basis data.