

2) INSTALASI & QUERY AWAL DATABASE

Instalasi MySQL

menggunakan termux

1. Buka termux
2. Ketik termux-setup-storage lalu Berikan akses termux ke memori internal
3. Muncul pop-up untuk meminta izin akses ke memori internal klik izinkan/allow acces
4. Lakukan update dan sekaligus upgrade paket dengan mengetik **pkg update && upgrade** lalu klik y
5. Jika ada konfirmasi untuk melanjutkan instalasi. Silahkan **klik y dan enter**
6. Instal aplikasi mariadb dengan mengetik **pkg install mariadb**
7. Memberikan akses aman ke MySQL dengan mengetik **mysql_safe**
8. Hentikan proses dengan **CTRL+Z**
9. Masuk kedalam admin dengan mengetik **MySQL -u root**

referensi video YouTube

<https://youtu.be/ez3nx3xH-y4?si=T4saycipqfBcqL1c>

Penggunaan awal MySQL

Query/struktur

```
MySQL -u root
```

Hasil

```
~ $ mysql -u root
mysql: Deprecated program name. It will be removed in a
future release, use '/data/data/com.termux/files/usr/bin
/mariadb' instead
Welcome to the MariaDB monitor.  Commands end with ; or
\g.
Your MariaDB connection id is 5
Server version: 11.1.2-MariaDB MariaDB Server

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

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

MariaDB [(none)]> 
```

Analisis

MySQL adalah perintah untuk mengakses Shell MySQL, yaitu antarmuka command line untuk berinteraksi dengan server MySQL

"-u root" adalah opsi yang digunakan saat Anda mengakses MySQL dari baris perintah (command line). Di sini, "-u" adalah singkatan dari "user", dan "root" adalah nama pengguna. Dengan menggunakan "-u root", Anda masuk ke MySQL sebagai pengguna "root", yang biasanya memiliki hak akses penuh ke semua fungsi MySQL. Pengguna "root" sering digunakan untuk melakukan tugas administratif dan konfigurasi yang memerlukan tingkat akses tertinggi.

Kesimpulan

MySQL -u root adalah bahwa Anda mengakses server MySQL dengan menggunakan opsi "-u" untuk menentukan pengguna (dalam hal ini, "root" yang merupakan pengguna dengan hak akses tertinggi). Dengan menggunakan "root", Anda memiliki kemampuan untuk melakukan tugas administratif dan konfigurasi yang memerlukan tingkat akses penuh ke server MySQL.

Database

CREATE DATABASE

Struktur :

```
CREATE DATABASE [nama_database];
```

Contoh :

```
CREATE DATABASE Taufiq;
```

Hasil

```
MariaDB [(none)]> CREATE DATABASE taufiq;  
Query OK, 1 row affected (0.002 sec)
```

```
MariaDB [(none)]> SHOW DATABASES;
```

```
+-----+  
| Database |  
+-----+  
| information_schema |  
| mysql |  
| performance_schema |  
| rental_taufiq |  
| sys |  
| taufiq |  
| test |  
| xi_rpl_1 |  
+-----+
```

```
8 rows in set (0.002 sec)
```

Analisis :

- `CREATE DATABASE` adalah sebuah query yang memerintahkan untuk membuat database.
- `taufiq` adalah nama dari suatu database yang kita buat

Kesimpulan

query atau kode tersebut memerintahkan untuk membuat suatu database yang bernama `taufiq`

TAMPILKAN DATABASE

Struktur :

```
SHOW DATABASES;
```

Contoh :

```
SHOW DATABASES;
```

Hasil :

```
MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| rental_taufiq   |
| sys            |
| taufiq          |
| test           |
| xi_rpl_1        |
+-----+
8 rows in set (0.002 sec)
```

Analisis :

- `SHOW DATABASE` merupakan query untuk menampilkan sebuah database dalam MySQL

Kesimpulan

`SHOW DATABASE` Suatu kode atau query yang berguna untuk menampilkan database yang sudah dibuat

HAPUS DATABASE

Struktur

```
DROP DATABASE [nama_database];
```

Contoh


```
DROP DATABASE [xi_rpl_1];
```

Hasil :

```
MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| rental_taufiq |
| sys |
| taufiq |
| test |
| xi_rpl_1 |
+-----+
8 rows in set (0.002 sec)

MariaDB [(none)]> DROP DATABASE taufiq;
Query OK, 0 rows affected (0.074 sec)

MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| rental_taufiq |
| sys |
| test |
| xi_rpl_1 |
+-----+
7 rows in set (0.008 sec)
```



Analisis

- `DROP DATABASE` Merupakan perintah query untuk menghapus database
- `taufiq` nama dari database yang ingin kita hapus

Kesimpulan

Untuk menghapus database kita bisa gunakan perintah `DROP DATABASE`

GUNAKAN DATABASE

Struktur

```
USE [nama_database];
```

Contoh

```
USE xi_rpl_1;
```

Hasil

```
MariaDB [(none)]> USE xi_rpl_1;  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A  
  
Database changed  
MariaDB [xi_rpl_1]> █
```

Analisis

- `USE` adalah perintah untuk masuk kedalam database
- `xi_rpl_1` merupakan nama database yang ingin kita gunakan atau masuki

Kesimpulan

Jika ingin masuk atau gunakan database, maka kita gunakan `USE` sebagai query

Tipe Data

ANGKA

- **INT**: Untuk menyimpan nilai bilangan bulat (integer). Misalnya, INT dapat digunakan untuk menyimpan angka seperti 1, 100, -10, dan sebagainya.
- **DECIMAL**: Digunakan untuk menyimpan nilai desimal presisi tinggi, cocok untuk perhitungan finansial atau keuangan.
- **FLOAT** dan **DOUBLE**: Digunakan untuk menyimpan nilai desimal dengan presisi floating-point. DOUBLE memiliki presisi lebih tinggi dibandingkan FLOAT.
- **TINYINT**, **SMALLINT**, **MEDIUMINT**, dan **BIGINT**: Tipe data ini menyimpan bilangan bulat dengan ukuran yang berbeda-beda.

Contoh

```
CREATE TABLE contoh_tabel (  
    id INT,  
    harga DECIMAL(10, 2),
```

```
    jumlah_barang TINYINT  
);
```

Hasil

```
MariaDB [xi_rpl_1]> show tables;  
+-----+  
| Tables_in_xi_rpl_1 |  
+-----+  
| contoh_tabel       |  
+-----+  
1 row in set (0.001 sec)  
  
MariaDB [xi_rpl_1]> desc contoh_tabel;  
+-----+-----+-----+-----+-----+-----+  
| Field          | Type          | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| id             | int(11)       | YES  |     | NULL    |       |  
| harga          | decimal(10,2) | YES  |     | NULL    |       |  
| jumlah_barang | tinyint(4)    | YES  |     | NULL    |       |  
+-----+-----+-----+-----+-----+-----+  
3 rows in set (0.006 sec)  
  
MariaDB [xi_rpl_1]> █
```

Kesimpulan

Dalam contoh tersebut, **id** menggunakan tipe data **INT**, **harga** menggunakan tipe data **DECIMAL** dengan presisi 10 digit dan 2 angka di belakang koma, dan **jumlah_barang** menggunakan tipe data **TINYINT**

TEXT

- **==CHAR(N) ==**Menyimpan string karakter tetap dengan panjang N. Contoh:
==CHAR(10) ==akan menyimpan string dengan panjang tepat 10 karakter.
- **VARCHAR(N):** Menyimpan string karakter dengan panjang variabel maksimal N.
Misalnya, **==VARCHAR(255) ==**dapat menyimpan string hingga 255 karakter, tetapi sebenarnya hanya menyimpan panjang yang diperlukan plus beberapa overhead.
- **==TEXT: ==**Digunakan untuk menyimpan teks dengan panjang variabel, tanpa batasan panjang tertentu. Cocok untuk data teks yang panjangnya tidak terduga.

Contoh

```
CREATE TABLE adiguna_tabel (  
    nama CHAR(50),
```

```
alamat VARCHAR(100),
catatan TEXT,
);
```

Hasil

```
MariaDB [xi_rpl_1]> SHOW TABLES;
```

```
+-----+
| Tables_in_xi_rpl_1 |
+-----+
| Taufiq_tabel       |
| adiguna_tabel       |
| contoh_tabel       |
+-----+
```

```
3 rows in set (0.001 sec)
```

```
MariaDB [xi_rpl_1]> DESC adiguna_tabel;
```

```
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| nama  | char(50)      | YES  |     | NULL    |       |
| alamat | varchar(100)  | YES  |     | NULL    |       |
| catatan | text         | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
```

```
3 rows in set (0.009 sec)
```

Kesimpulan

Dalam contoh tersebut, **nama** menggunakan tipe data **char** dengan panjang tetap, **alamat** menggunakan tipe data **VARCHAR** dengan panjang variabel, **catatan** menggunakan tipe data **TEXT** untuk menyimpan teks yang mungkin panjangnya bervariasi, dan **status** menggunakan tipe data **ENUM** untuk membatasi nilai yang mungkin.

TANGGAL

- **DATE** : Menyimpan nilai tanggal dengan format YYYY-MM-DD.
- **TIME** : Menyimpan nilai waktu dengan format HH:MM:SS.
- **DATETIME** : Menggabungkan nilai tanggal dan waktu dengan format YYYY-MM-DD HH:MM:SS
- **==TIMESTAMP**: ==Sama seperti DATETIME, tetapi dengan kelebihan diatur secara otomatis saat data dimasukkan atau diubah.

Contoh


```
CREATE TABLE Taufiq_tabel (  
    tanggal DATE,  
    waktu TIME,  
    datetimekolom DATETIME,  
    timestampkolom TIMESTAMP  
);
```

Hasil

```
MariaDB [xi_rpl_1]> show tables;
```

```
+-----+  
| Tables_in_xi_rpl_1 |  
+-----+  
| Taufiq_tabel  
| adiguna_tabel  
| contoh_tabel  
+-----+
```

```
3 rows in set (0.002 sec)
```

```
MariaDB [xi_rpl_1]> desc Taufiq_tabel;
```

```
+-----+-----+-----+-----+-----+-----+  
| Field          | Type      | Null    | Key    | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| tanggal       | date      | YES     |        | NULL    |       |  
| waktu          | time      | YES     |        | NULL    |       |  
| datetimekolom  | datetime  | YES     |        | NULL    |       |  
| timestampkolom | timestamp | YES     |        | NULL    |       |  
+-----+-----+-----+-----+-----+-----+
```

```
4 rows in set (0.010 sec)
```

Kesimpulan

Dalam contoh ini, kolom **tanggal** akan menyimpan nilai tanggal, **waktu** menyimpan nilai waktu, **datetimekolom** menyimpan kombinasi tanggal dan waktu, dan **timestampkolom** akan secara otomatis diatur saat data dimasukkan atau diubah.

BOOLEAN

Boolean adalah tipe data yang hanya memiliki dua nilai: true atau false. Ini digunakan untuk menyatakan kebenaran suatu kondisi atau pernyataan. Dalam banyak bahasa pemrograman, termasuk SQL, boolean digunakan untuk pengambilan keputusan dan pengendalian alur program.

```
CREATE TABLE contohTabel (  
    title VARCHAR(255),  
    completed BOOLEAN
```

```
);``
```

Dalam contoh diatas, kita mendefinisikan kolom `***completed***` sebagai tipe data `***BOOLEAN***`. Ini merupakan cara yang sah dan umum digunakan di MySQL. Nilai yang dapat disimpan dalam kolom ini adalah `***TRUE***` atau `***FALSE***`, atau dalam representasi angka, 1 atau 0.

==Tipe Data Pilihan ==

ENUM

Penjelasan

Dalam MySQL, ENUM adalah tipe data khusus yang memungkinkan Anda mendefinisikan sebuah kolom yang dapat mengandung satu dari beberapa nilai konstan yang telah ditentukan. Ini memberikan cara yang efisien untuk menyimpan dan memvalidasi nilai dalam kolom.

Contoh

```MySQL

```
CREATE TABLE ExampleTable (
 id INT PRIMARY KEY,
 status ENUM('Active', 'Inactive', 'Pending') NOT NULL
);
```

## Hasil

```
MariaDB [xi_rpl_1]> CREATE TABLE ExampleTable (
-> id INT PRIMARY KEY,
-> status ENUM('Active', 'Inactive', 'Pending') NOT NULL
->);
```

Query OK, 0 rows affected (0.119 sec)

```
MariaDB [xi_rpl_1]> desc ExampleTable;
```

| Field  | Type                                | Null | Key | Default | Extra |
|--------|-------------------------------------|------|-----|---------|-------|
| id     | int(11)                             | NO   | PRI | NULL    |       |
| status | enum('Active','Inactive','Pending') | NO   |     | NULL    |       |

2 rows in set (0.018 sec)

```
MariaDB [xi_rpl_1]>
```

### Kesimpulan

status adalah kolom dengan tipe data ENUM yang dapat berisi nilai 'Active', 'Inactive', atau 'Pending'. Dengan menggunakan ENUM, Anda membatasi kolom tersebut hanya menerima nilai-nilai yang telah ditentukan, mengurangi kemungkinan kesalahan input dan meningkatkan kejelasan dalam struktur data tabel.

# SET

tipe data SET digunakan untuk mendefinisikan sebuah kolom yang dapat memiliki satu atau beberapa nilai dari sekumpulan nilai yang telah ditentukan. Nilai-nilai dalam tipe data SET disimpan sebagai himpunan tanpa urutan tertentu, dan setiap nilai hanya dapat muncul satu kali.

## Contoh

```
CREATE TABLE Set_tabel(
 id INT PRIMARY KEY,
 preferences SET('Email', 'SMS') NOT NULL
);
```

## Hasil

```
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id | int(11) | NO | PRI | NULL | |
| preferences | set('Email','SMS') | NO | | NULL | |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.011 sec)

MariaDB [xi_rpl_1]> █
```

### kesimpulan

Preferences ada kolom dengan tipe data set yang terdapat 2 pilihan didalam nya yaitu Email dan SMS

## TABLE

## BUAT TABLE

### Struktur

```
CREATE TABLE nama_tabel (
 Namakolom typedata(lebar) cons,
 Namakolom typedata(lebar) cons,
 Namakolom typedata(lebar) cons,
);
```

## Contoh :

```
Create table daftar_siswa;
Id int primary key,
Nama varchar(255),
Umur int,
Alamat text
);
```

## Hasil

```
MariaDB [rental_taufiq]> desc daftar_siswa;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
id	int(11)	NO	PRI	NULL	
nama	varchar(255)	YES		NULL	
umur	int(11)	YES		NULL	
alamat	text	YES		NULL	
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.094 sec)

MariaDB [rental_taufiq]> █
```

## Analisis :

- `CREATE TABLE` Digunakan untuk membuat tabel
- `daftar_siswa` nama dari tabel yang ingin kita buat
- `id` : Kolom untuk menyimpan ID sebagai kunci utama (Primary Key).
- `Primary key` merupakan kunci unik yg ada pada tabel dan hanya 1 setiap tabelnya
- `nama` : Kolom untuk menyimpan nama dengan tipe data VARCHAR (teks dengan panjang variabel).
- `umur` : Kolom untuk menyimpan umur dengan tipe data INT (bilangan bulat).
- `alamat` : Kolom untuk menyimpan alamat dengan tipe data TEXT (teks dengan panjang variabel).

### Kesimpulan

`CREATE TABLE daftar_siswa;` digunakan untuk membuat suatu tabel yg ingin digunakan untuk menampung daftar siswa sesuai dengan namanya yaitu `daftar_siswa`.

---

## TAMPILKAN STRUKTUR TABLE

## Struktur

```
DESC nama_tabel;
```

## Contoh Desc Tabel :

```
Desc daftar_siswa;
```

## Contoh Hasil :

```
MariaDB [rental_taufiq]> desc daftar_siswa;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
id	int(11)	NO	PRI	NULL	
nama	varchar(255)	YES		NULL	
umur	int(11)	YES		NULL	
alamat	text	YES		NULL	
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.094 sec)

MariaDB [rental_taufiq]> █
```

## Analisis Desc Tabel :

- `Desc` berfungsi memberikan informasi kolom-kolom dalam tabel yang di buat, termasuk nama kolom, tipe data,serta penggunaan null dalam kolom itu bisa terlihat di Desc Tabel.
- `Daftar_siswa;` merupakan sebuah nama tabel ingin di lihat strukturnya.

### kesimpulan

jika kita ingin melihat tipe data yang ada pada kolom-kolom tabel, kita dapat menampilkan nya dengan Desc [nama\_tabel]

---

## MENAMPILKAN DAFTAR TABLE

## Struktur

```
SHOW TABLES;
```

## Contoh Show tables :

```
SHOW TABLES;
```

## Hasil Show tables :

```
MariaDB [rental_taufiq]> show tables;
+-----+
| Tables_in_rental_taufiq |
+-----+
| daftar_siswa |
| pelanggan |
+-----+
```

## Analisis

- `SHOW TABLES` digunakan untuk memanggil tabel-tabel yang sudah dibuat dalam database rental\_taufiq
- dalam database rental\_taufiq terdapat 2 table yang bernama daftar\_siswa dan pelanggan

### kesimpulan

jika ingin mengetahui ada berapa tabel yang sudah dibuat dalam suatu database kita dapat menggunakan Query `SHOW TABLES ;`

## QNA

 1

Mengapa hanya kolom id\_pelanggan yang menggunakan constraint primary key?

### jawaban

Karna hanya kolom id\_pelanggan yang memiliki id unik, yang tidak ada samanya dengan baris lain dalam tabel, sehingga kita pilih id\_pelanggan sebagai primary key

 2

mengapa pada kolom no\_telp yang menggunakan tipe data char bukan varchar?

### Jawaban

Karna tipe data char merupakan tipe data yang menetapkan panjang karakter, seperti yang di ketahui no\_telp memiliki panjang yang tetap maka dari itu kita menggunakan tipe data char, karna jika tipe data varchar digunakan untuk text yang panjang

🔍 3

mengapa hanya kolom no\_telp yang menggunakan constraint unique?

### Jawaban

karna setiap pelanggan mempunyai no\_telp masing" yang artinya nomornya pasti berbeda, maka dari itu kita pake unique agar setiap pelanggan memasukkan nomor telepon nya sendiri

🔍 4

mengapa kolom no\_telp tidak memakai constraint Not null, sementara kolom lainnya menggunakan constraint tersebut?

### Jawaban

berarti dalam tabel tersebut no\_telp hanya opsional saja dan tidak wajib di isi/bisa dikosongkan

🔍 5

perbedaan primary key dan unique

### Jawaban

Dalam `primary key` hanya boleh 1 setiap tabelnya, sedangkan unique bisa lebih dari 1.

`primary key` bertujuan mengidentifikasi secara unik setiap baris dalam tabel, sedangkan `unique` Berfungsi untuk memastikan bahwa tidak ada dua baris dalam tabel yang memiliki nilai yang sama pada kolom yang memiliki constraint `UNIQUE`.

---

## Insert

### INSERT 1 DATA

#### Struktur

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

#### Contoh

```
INSERT INTO pelanggan
VALUES (123,"TAUFIQ","ADIGUNA","1234567890");
```

#### Hasil :

```
+-----+-----+-----+-----+
| id | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
| 123 | TAUFIQ | ADIGUNA | 1234567890 |
+-----+-----+-----+-----+
```

#### Analisis

- `INSERT INTO` Query awal untuk menambahkan data di dalam tabel
- `pelanggan` merupakan nama tabel yang ingin kita masukkan data kedalamnya
- `VALUES (123,"TAUFIQ","ADIGUNA","1234567890");` merupakan nilai yang ingin kita masukkan kedalam kolom-kolom yang sudah ada pada tabel pelanggan

#### Kesimpulan

Jika ingin menambahkan nilai dalam tabel, kita bisa menggunakan perintah `INSERT`

---

### INSERT >1 DATA

#### Struktur

```
INSERT INTO [nama_table]
VALUES (kolom1,kolom2,kolom3,kolom4),
 (kolom1,kolom2,kolom3,kolom4),
```



```
(kolom1,kolom2,kolom3,kolom4),
```

## Contoh

```
INSERT INTO pelanggan
VALUES
 (124, "ALFAHREZI", "RAIHAN", "0812345678"),
 (125, "FARID", "WIBOWO", "9876543210"),
 (126, "SITI", "RAHMA", "5678901234");
```

## Hasil

```
+-----+-----+-----+-----+
| id | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
123	TAUFIQ	ADIGUNA	1234567890
124	ALFAHREZI	RAIHAN	0812345678
125	FARID	WIBOWO	9876543210
126	SITI	RAHMA	5678901234
+-----+-----+-----+-----+
4 rows in set (0.003 sec)
```

## Analisis

- `INSERT INTO` Query awal untuk menambahkan data di dalam tabel
- `pelanggan` merupakan nama tabel yang ingin kita masukkan data kedalamnya
- `VALUES`

```
(124, "ALFAHREZI", "RAIHAN", "0812345678"
(125, "FARID", "WIBOWO", "9876543210"),
(126, "SITI", "RAHMA", "5678901234");
```

Nilai yang akan kita masukkan kedalam tabel lebih dari 1 data.

### Kesimpulan

jika ingin menambahkan data lebih dari 1 kita bisa menggunakan metode seperti ini

## MENYEBUT KOLOM

### Struktur

```
INSERT INTO [nama_tabel]
(Kolom1,kolom2,kolom3,..)
```

```
VALUES (nilai1,nilai2,nilai3,...)
```

## Contoh

```
INSERT INTO pelanggan
 (id,nama_depan,nama_belakang)
VALUES (127,"ZHAFRAN","RIZKI");
```

## Hasil

```
MariaDB [rental_taufiq]> INSERT INTO pelanggan
-> (id,nama_depan,nama_belakang)
-> VALUES (127,"ZHAFRAN","RIZKI");
Query OK, 1 row affected (0.062 sec)

MariaDB [rental_taufiq]> SELECT * FROM pelanggan;
+-----+-----+-----+-----+
| id | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
123	TAUFIQ	ADIGUNA	1234567890
124	ALFAHREZI	RAIHAN	0812345678
125	FARID	WIBOWO	9876543210
126	SITI	RAHMA	5678901234
127	ZHAFRAN	RIZKI	NULL
+-----+-----+-----+-----+
5 rows in set (0.006 sec)
```

## Analisis

- `INSERT INTO` Query awal untuk menambahkan data di dalam tabel
- `pelanggan` merupakan nama tabel yang ingin kita masukkan data kedalamnya
- `(id,nama_depan,nama_belakang)` merupakan kolom pilihan yang ingin kita tambahkan data kedalamnya
- `VALUES (127,"ZHAFRAN","RIZKI");` data yang akan kita masukkan kedalam tabel sesuai dengan kolom yang sudah di tentukan

### Kesimpulan

Untuk menambahkan data di kolom tertentu kita harus memanggil kolom yang ingin kita tambahkan data didalamnya terlebih dahulu Setelah itu data-data apa yang ingin kita masukkan

---

## SELECT

## SELURUH DATA

## Struktur

```
SELECT * FROM [nama_tabel];
```

## Contoh

```
SELECT * FROM pelanggan;
```

## Hasil

```
MariaDB [rental_taufiq]> SELECT * FROM pelanggan;
+-----+-----+-----+-----+
| id | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
123	TAUFIQ	ADIGUNA	1234567890
124	ALFAHREZI	RAIHAN	0812345678
125	FARID	WIBOWO	9876543210
126	SITI	RAHMA	5678901234
127	ZHAFRAN	RIZKI	NULL
+-----+-----+-----+-----+
5 rows in set (0.014 sec)
```

## Analisis

- `SELECT` digunakan untuk mengambil/menampilkan Data pada tabel database.
- `*` digunakan jika ingin mengambil/menampilkan semua kolom
- `FROM pelanggan` Merupakan nama tabel yang ingin di tampilkan

### Kesimpulan

untuk mengambil semua data di tabel, kita bisa menggunakan `Query` dengan kata kunci `SELECT * FROM`

---

## DATA KOLOM TERTENTU

### Struktur

```
SELECT [nama_kolom1], [nama_kolom2], [nama_kolom3],... [nama_kolom_n],
FROM [nama_tabel];
```

### Contoh

```
SELECT nama_depan,no_telp FROM pelanggan;
```

## Hasil

```
MariaDB [rental_taufiq]> SELECT nama_depan,no_telp FROM pelanggan;
+-----+-----+
| nama_depan | no_telp |
+-----+-----+
TAUFIQ	1234567890
ALFAHREZI	0812345678
FARID	9876543210
SITI	5678901234
ZHAFRAN	NULL
+-----+-----+
5 rows in set (0.007 sec)
```

## Analisis

- `SELECT nama_depan,no_telp` query untuk menampilkan kolom nama depan dan no telp
- `FROM pelanggan` merupakan nama tabel yang ingin kita tampilkan kolom nama depan dan no telp

### Kesimpulan

Jika ingin menampilkan kolom-kolom tertentu atau yang di inginkan kita bisa gunakan perintah `SELECT nama_kolom FROM pelanggan`

---

## KLAUSA WHERE

### Struktur

```
SELECT * FROM [nama_tabel]
WHERE [nama_kolom] = [nilai];
```

### Format kondisi

nama kolom : seperti id,nama\_depan,no\_telp  
operator : seperti =,>,>=,<,<=,!<>,dst  
nilai : seperti pada id 123

## Contoh

```
SELECT * FROM pelanggan
WHERE id=123;
```

## Hasil

```
MariaDB [rental_taufiq]> SELECT * FROM pelanggan
-> WHERE id=123;
+-----+-----+-----+-----+
| id | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
| 123 | TAUFIQ | ADIGUNA | 1234567890 |
+-----+-----+-----+-----+
1 row in set (0.002 sec)
```

## Analisis

- `SELECT * FROM pelanggan` merupakan query untuk menampilkan tabel pelanggan
- `WHERE id=123;` adalah perintah untuk menampilkan sebuah baris yang memiliki id 123

### Kesimpulan

Jika ingin mencari data tertentu yang ada pada tabel kita bisa gunakan `WHERE`  
`[nama_kolom] = [nilai]`

## Update

## Struktur

```
UPDATE [nama_tabel]
SET [nama_kolom] = "nilai"
WHERE [nama_kolom] = "nilai";
```

## Contoh

```
UPDATE pelanggan
Set no_telp = "085340305553"
WHERE id=123;
```

## Hasil

```
MariaDB [rental_taufiq]> SELECT * FROM pelanggan;
```

| id  | nama_depan | nama_belakang | no_telp    |
|-----|------------|---------------|------------|
| 123 | TAUFIQ     | ADIGUNA       | 8534030552 |
| 124 | ALFAHREZI  | RAIHAN        | 0812345678 |
| 125 | FARID      | WIBOWO        | 9876543210 |
| 126 | SITI       | RAHMA         | 5678901234 |
| 127 | ZHAFRAN    | RIZKI         | NULL       |

```
5 rows in set (0.002 sec)
```

```
MariaDB [rental_taufiq]> UPDATE pelanggan
-> SET no_telp = "085340305553"
-> WHERE id=123;
```

```
Query OK, 1 row affected (0.003 sec)
```

```
Rows matched: 1 Changed: 1 Warnings: 0
```

```
MariaDB [rental_taufiq]> SELECT * FROM pelanggan;
```

| id  | nama_depan | nama_belakang | no_telp      |
|-----|------------|---------------|--------------|
| 123 | TAUFIQ     | ADIGUNA       | 085340305553 |
| 124 | ALFAHREZI  | RAIHAN        | 0812345678   |
| 125 | FARID      | WIBOWO        | 9876543210   |
| 126 | SITI       | RAHMA         | 5678901234   |
| 127 | ZHAFRAN    | RIZKI         | NULL         |

```
5 rows in set (0.001 sec)
```

## Analisis

- `Update` merupakan query untuk mengupdate data dalam kolom
- `pelanggan` adalah nama dari database yang ingin di ubah
- `SET` digunakan untuk mengubah nilai kolom pada sebuah baris data.
- `no_telp = "085340305553"` `no_telp` nama Kolom yang ingin di ubah nilainya menjadi `085340305553` sebagai nilai yang baru

jika ingin mengganti atau update data pada kolom kita bisa gunakan query `UPDATE`  
`nama_table SET nama_kolom = value`

## DELETE BARIS

### Struktur

```
DELETE FROM [nama_tabel]
WHERE [nama_kolom] = [nilai];
```

### Contoh

```
DELETE FROM pelanggan
WHERE id = "127";
```

### Hasil

```
MariaDB [rental_taufiq]> DELETE FROM pelanggan
-> WHERE id="127";
Query OK, 1 row affected (0.005 sec)

MariaDB [rental_taufiq]> SELECT * FROM pelanggan;
+-----+-----+-----+-----+
| id | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
123	TAUFIQ	ADIGUNA	085340305553
124	ALFAHREZI	RAIHAN	0812345678
125	FARID	WIBOWO	9876543210
126	SITI	RAHMA	5678901234
+-----+-----+-----+-----+
4 rows in set (0.001 sec)
```

### Analisis

- `DELETE` Merupakan perintah query untuk menghapus
- `FROM PELANGGAN` Merupakan nama tabel yang ingin kita hapus datanya
- `WHERE id="127";` Query untuk menghapus baris pada tabel pelanggan yang memiliki `id=127`

Jika ingin menghapus suatu baris tertentu pada tabel kita dapat gunakan query `DELETE FROM [nama_tabel] WHERE [nama_kolom] = [nilai];`

## DELETE TABEL

### Struktur

```
DROP TABLE [nama_tabel];
```

### Contoh

```
DROP TABLE penerima_pip;
```

### Hasil

```
MariaDB [rental_taufiq]> SHOW TABLES;
+-----+
| Tables_in_rental_taufiq |
+-----+
| daftar_siswa |
| pelanggan |
| penerima_pip |
+-----+
3 rows in set (0.004 sec)

MariaDB [rental_taufiq]> DROP TABLE penerima_pip;
Query OK, 0 rows affected (0.008 sec)

MariaDB [rental_taufiq]> SHOW TABLES;
+-----+
| Tables_in_rental_taufiq |
+-----+
| daftar_siswa |
| pelanggan |
+-----+
2 rows in set (0.002 sec)
```

### Analisis

- `DROP TABLE` Merupakan query untuk menghapus sebuah tabel.
- `penerima_pip` nama tabel yang ingin kita hapus.



## kesimpulan

ketika ingin menghapus sebuah tabel dalam database, kita dapat gunakan perintah query `DROP TABLE [nama_tabel];`