

Instalasi MySQL

Menggunakan Xampp/Termux

Termux

1. Buka Termux
2. Berikan akses Termux ke memori Internal `termux-setup-storage`
3. Muncul Pop-up untuk meminta izin akses ke memori internal "klik di izinkan/allow acces"
4. Lakukan Update dan sekaligus upgrade paket `pkg update && upgrade -y`
5. Jika ada konfirmasi untuk melanjutkan instalasi. Silahkan "klik y" dan "enter"
6. Instal aplikasi mariadb `pkg install mariadb`
7. Memberikan Akses Aman ke MySQL `mysqld_safe`
8. Hentikan Proses `Ctrl Z`
9. Masuk kedalam Admin `mysql -u root`

Referensi Video YouTube

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

Penggunaan Awal MySQL

Query

```
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 :

Ini adalah perintah untuk mengakses shell **MySQL**, yaitu antarmuka command-line untuk berinteraksi dengan server **MySQL**.

-u root :

Parameter ini menentukan **pengguna (user)** yang akan digunakan untuk masuk ke server MySQL. Dalam hal ini "**root**" Adalah nama pengguna yang diberikan, dan "**root**" Adalah tingkat tertinggi dengan hak akses penuh.

Kesimpulan

Kesimpulan :

`mysql -u root` memberikan akses penuh ke server **MySQL** dengan menggunakan pengguna "**root**", yang memiliki hak akses maksimum. Penggunaan perintah ini perlu hati-hati untuk menghindari risiko keamanan

Database

Buat Database

Untuk membuat database di SQL, Anda dapat menggunakan perintah `CREATE DATABASE` dengan menentukan nama database yang diinginkan. Pastikan untuk memiliki hak akses yang sesuai, dan verifikasi pembuatan database dengan perintah `SHOW DATABASES` .

Pengetahuan tentang sintaks SQL dan hak akses server diperlukan untuk menjalankan operasi ini dengan sukses.


STRUKTUR

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

CONTOH

```
CREATE DATABASE xi_rpl_1;
```

HASIL

```
MariaDB [(none)]> create database xi_rpl_1;   
Query OK, 1 row affected (0.006 sec)
```

```
MariaDB [(none)]> show databases;
```

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

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

```
MariaDB [(none)]> █
```

ANALISIS

Kode tersebut merupakan perintah untuk membuat database baru dengan nama "xi_rpl_1".

KESIMPULAN

Kesimpulan dari kode tersebut adalah bahwa perintah tersebut bertujuan untuk membuat sebuah database baru dengan nama "xi_rpl_1".

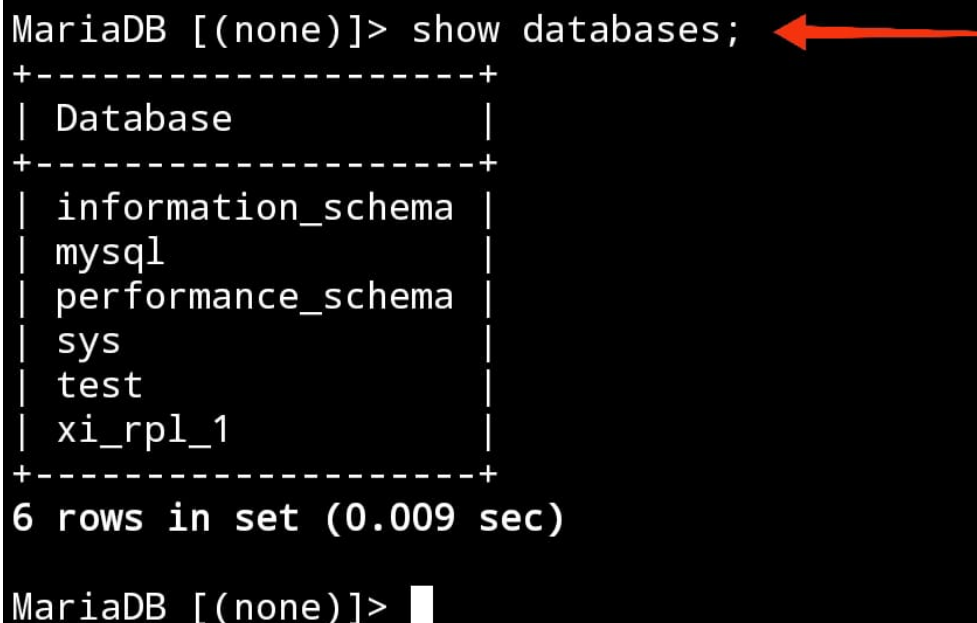
Tampilkan Database

Untuk menampilkan daftar database di MySQL, Anda dapat menggunakan perintah SQL `SHOW DATABASES;`. Perintah ini memberikan gambaran keseluruhan database yang tersedia di server MySQL. Pastikan pengguna yang digunakan memiliki izin untuk melihat database, dan gunakan perintah ini melalui antarmuka command-line atau alat manajemen database seperti phpMyAdmin.

KODENYA

```
SHOW DATABASES;
```

HASIL



```
MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| sys            |
| test           |
| xi_rp1_1       |
+-----+
6 rows in set (0.009 sec)

MariaDB [(none)]> 
```

ANALISIS

Kode tersebut merupakan perintah untuk menampilkan daftar database yang ada dalam sistem database.

KESIMPULAN

Kode `SHOW DATABASES;` digunakan untuk menampilkan daftar semua database yang ada dalam sistem basis data yang sedang digunakan. Dengan demikian, kesimpulannya adalah perintah ini bertujuan untuk memberikan informasi tentang semua database yang telah dibuat atau tersedia.

Hapus Database

Untuk menghapus sebuah database di SQL, Anda dapat menggunakan perintah `DROP DATABASE`. Namun, perlu diingat bahwa tindakan ini permanen dan akan menghapus seluruh data di dalam database tersebut. Pastikan Anda memiliki backup data yang dibutuhkan sebelum melanjutkan.

STRUKTUR

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

CONTOH

```
DROP DATABASE xi_rpl_1;
```

HASIL

```
MariaDB [(none)]> show databases;
```

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

```
6 rows in set (0.014 sec)
```

```
MariaDB [(none)]> drop database xi_rpl_1;  
Query OK, 0 rows affected (0.013 sec)
```



```
MariaDB [(none)]> show databases;
```

```
+-----+  
| Database |  
+-----+  
| information_schema |  
| mysql |  
| performance_schema |  
| sys |  
| test |  
+-----+
```

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

```
MariaDB [(none)]> █
```

ANALISIS

Kode `DROP DATABASE xi_rpl_1;` digunakan untuk menghapus database dengan nama "xi_rpl_1". Perlu diperhatikan bahwa perintah ini bersifat permanen dan akan menghapus semua data yang terkait dengan database tersebut.

KESIMPULAN

Kesimpulan dari kode `DROP DATABASE xi_rpl_1;` adalah bahwa perintah tersebut bertujuan untuk menghapus database permanen dengan nama "xi_rpl_1" beserta seluruh data yang terkait.

Gunakan Database

perintah `USE` digunakan untuk beralih atau menggunakan sebuah database tertentu di server. Perintah ini sangat berguna ketika Anda bekerja dengan beberapa database di

server MySQL dan ingin fokus pada satu database dalam sesi tertentu.

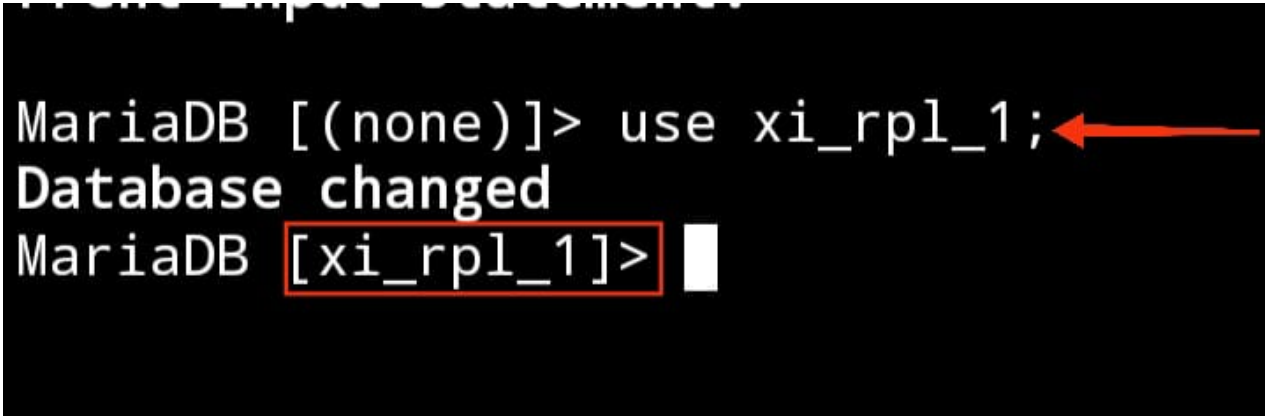
STRUKTUR

```
USE [nama_database];
```

CONTOH

```
USE xi_rpl_1;
```

HASIL



The screenshot shows a MySQL command prompt with a black background and white text. The prompt starts with 'MariaDB [(none)]>'. The user enters 'use xi_rpl_1;', which is followed by a red arrow pointing to the semicolon. The prompt then changes to 'Database changed'. Finally, the prompt shows 'MariaDB [xi_rpl_1]>' where the database name 'xi_rpl_1' is enclosed in a red rectangular box.

ANALISIS

Kode `USE xi_rpl_1;` digunakan untuk beralih dan menggunakan database dengan nama "xi_rpl_1". Ini menetapkan database tersebut sebagai database aktif, sehingga perintah-perintah selanjutnya akan berlaku untuk database tersebut.

KESIMPULAN

Kesimpulan dari kode `USE xi_rpl_1;` adalah bahwa perintah tersebut bertujuan untuk beralih dan menggunakan database aktif dengan nama "xi_rpl_1".

Tugas 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 PROGRAM :

```
MariaDB [xi_rpl_1]> SHOW TABLES;
+-----+
| Tables_in_xi_rpl_1 |
+-----+
| contoh_tabel       |
+-----+
1 row in set (0.003 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 (0.028 sec)

MariaDB [xi_rpl_1]> █
```

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**.

Teks

- **==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 farel_tabel (
    nama CHAR(50),
    alamat VARCHAR(100),
    catatan TEXT,
    status ENUM('Aktif', 'Non-Aktif')
);
```

HASIL PROGRAM :

```
MariaDB [xi_rpl_1]> SHOW TABLES;
+-----+
| Tables_in_xi_rpl_1 |
+-----+
| contoh_tabel       |
| farel_tabel        |
+-----+
2 rows in set (0.002 sec)

MariaDB [xi_rpl_1]> DESC farel_tabel;
+-----+-----+-----+-----+-----+-----+
| Field | Type                               | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| nama  | char(50)                          | YES  |     | NULL    |       |
| alamat | varchar(100)                      | YES  |     | NULL    |       |
| catatan | text                             | YES  |     | NULL    |       |
| status | enum('Aktif','Non-Aktif')         | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.007 sec)

MariaDB [xi_rpl_1]> 
```

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 frel_Tabel (
    tanggal DATE,
    waktu TIME,
    datetimekolom DATETIME,
    timestampkolom TIMESTAMP
);
```

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 / TINYINT(1)**: Digunakan untuk menyimpan nilai boolean, yang dapat mewakili kebenaran atau kesalahan. Representasi nilai benar adalah 1, sedangkan nilai salah direpresentasikan sebagai 0. Meskipun nilai selain 0 dianggap benar, secara umum, ketiganya seringkali digunakan secara bergantian. Seringkali, ketika Anda mendeklarasikan kolom sebagai BOOL atau BOOLEAN, MySQL mengonversinya secara otomatis menjadi TINYINT(1), yang juga dapat digunakan untuk menyimpan nilai boolean dengan 0 untuk false dan 1 untuk true.

Menggunakan BOOLEAN

```
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.

Menggunakan BOOL

```
``MySQL
CREATE TABLE contohTabel (
    title VARCHAR(255),
```

```
completed BOOL  
);
```

Dalam contoh ini, kita menggunakan `BOOL` sebagai tipe data untuk kolom `completed`. Perlu dicatat bahwa MySQL secara otomatis mengonversi `BOOL` menjadi `TINYINT(1)`. Oleh karena itu, pada dasarnya, ini setara dengan contoh pertama. Namun, beberapa pengembang lebih suka menggunakan `BOOLEAN` untuk kejelasan.

Menggunakan TINYINT(1)

```
CREATE TABLE contohTabel (  
    title VARCHAR(255),  
    completed TINYINT(1)  
);
```

Dalam contoh ini, kita menggunakan `TINYINT(1)` sebagai tipe data untuk kolom `completed`. Ini adalah pendekatan yang valid karena MySQL mengonversi `BOOL` menjadi `TINYINT(1)` secara otomatis. Dalam hal ini, nilai yang dapat disimpan adalah 1 untuk `TRUE` dan 0 untuk `FALSE`.

Tipe Data Pilihan

- `==ENUM:` ==Memungkinkan Anda mendefinisikan set nilai yang mungkin dan membatasi kolom hanya dapat mengambil salah satu dari nilai tersebut.
- `==SET:` ==Mirip dengan `ENUM`, namun dapat menyimpan satu atau lebih nilai dari himpunan yang telah ditentukan.

Tabel

Buat Tabel

STRUKTUR

```
CREATE TABLE nama_tabel (  
    kolom1 tipe_data(max karakter) Constraint: kunci  
    induk constraint: data tidak boleh kosong ,  
    kolom2 tipe_data(max karakter) constraint: data tidak boleh kosong ,  
    kolom3 tipe_data(max karakter) ,  
    kolom4 tipe_data(max karakter) constraint: tidak ada data yang sama );
```

CONTOH

```
CREATE TABLE pelanggan ( id_pelanggan int(4)PRIMARY KEY NOT NULL ,
nama_depan varchar(25) NOT NULL , nama_belakang varchar(25), no_telp
char(12)UNIQUE );
```

HASIL

```
MariaDB [rental_farel]> 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)	YES		NULL	
no_telp	char(12)	YES	UNI	NULL	

```
4 rows in set (0.005 sec)
```

```
MariaDB [rental_farel]> █
```

ANALISIS

`CREATE TABLE` : diikuti oleh nama tabel yang ingin dibuat, yaitu "pelanggan". Setelah itu, di dalam tanda kurung, diberikan definisi untuk setiap kolom yang akan dimiliki oleh tabel tersebut. Setiap kolom memiliki nama, tipe data, dan batasan-batasan tertentu.

`id_pelanggan` : Kolom ini memiliki tipe data integer `INT` dengan panjang maksimum 4 digit. Kolom ini juga ditetapkan sebagai kunci utama (`PRIMARY KEY`) yang berarti nilainya harus unik untuk setiap baris dalam tabel dan tidak boleh kosong (`NOT NULL`).

`nama_depan` : Kolom ini memiliki tipe data `varchar` dengan panjang maksimum 25 karakter. Kolom ini juga ditetapkan sebagai kolom yang tidak boleh kosong (`NOT NULL`).

`nama_belakang` : Kolom ini memiliki tipe data `varchar` dengan panjang maksimum 25 karakter. Kolom ini dibiarkan opsional, yang berarti nilai-nilainya boleh kosong.

`no_telp` : Kolom ini memiliki tipe data `char` dengan panjang 12 karakter. Kolom ini ditetapkan sebagai kolom yang harus memiliki nilai unik (`UNIQUE`), yang berarti setiap nomor telepon harus unik di antara semua entri dalam tabel.

KESIMPULAN

Kesimpulan dari kode tersebut adalah bahwa perintah tersebut digunakan untuk membuat tabel baru bernama "pelanggan" dengan beberapa kolom seperti "id_pelanggan", "nama_depan", "nama_belakang", dan "no_telp". Kolom "id_pelanggan" diatur sebagai kunci utama yang tidak boleh kosong, "nama_depan" dan "no_telp" juga tidak boleh kosong, dan kolom "no_telp" harus memiliki nilai yang unik.

Tampilkan Struktur Tabel

STRUKTUR

```
DESC nama_tabel;
```

CONTOH

```
DESC pelanggan;
```

HASIL

```
MariaDB [rental_farel]> 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)   | YES  |     | NULL    |       |
| no_telp        | char(12)      | YES  | UNI | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.005 sec)

MariaDB [rental_farel]> █
```

ANALISIS

Kode `DESC pelanggan;` digunakan untuk mendapatkan deskripsi atau struktur dari tabel "pelanggan". Ini memberikan informasi tentang kolom-kolom dalam tabel beserta tipe data, dan constraint (jika ada).

KESIMPULAN

Kesimpulan dari kode `DESC pelanggan;` adalah bahwa perintah tersebut bertujuan untuk menampilkan deskripsi atau struktur dari tabel dengan nama "pelanggan". Ini memberikan informasi tentang kolom-kolom yang ada dalam tabel, seperti nama kolom, tipe data, dan konstrain yang mungkin diterapkan.

Tampilkan Daftar Tabel

STRUKTUR

```
SHOW TABLES;
```

CONTOH

```
SHOW TABLES;
```

HASIL

```
Database changed
MariaDB [rental_fare1]> show Tables;
+-----+
| Tables_in_rental_fare1 |
+-----+
| pelanggan              |
+-----+
1 row in set (0.002 sec)
```

ANALISIS

Kode `SHOW TABLES;` digunakan untuk menampilkan daftar semua tabel yang ada dalam database yang sedang digunakan. Ini membantu pengguna untuk melihat tabel-tabel mana yang telah dibuat atau tersedia dalam database tersebut.

KESIMPULAN

Kesimpulan dari `SHOW TABLES;` adalah perintah tersebut digunakan untuk menampilkan daftar semua tabel yang ada dalam database yang sedang digunakan.

QnA

[🔗 Perbedaan antara PRIMARY KEY dan UNIQUE >](#)

PRIMARY KEY:

Mirip seperti nomor identitas yang unik bagi setiap baris, tak boleh kosong.

UNIQUE:

Lebih seperti nomor yang bisa digunakan untuk keperluan lain, harus unik tapi bisa juga kosong.

PRIMARY KEY :

bertugas membedakan nilai yang ada pada tabel seperti NIS.

UNIQUE :

bertugas untuk memastikan bahwa tidak ada nilai duplikat dalam kolom tersebut. contohnya seperti, no wa, dan alamat email.

❓ Mengapa Hanya Kolom Id Pelanggan yang menggunakan Constraint "PRIMARY KEY"?



Karena kolom ID pelanggan berperan sebagai identitas unik untuk setiap entitas pelanggan dalam basis data.

Dapat memastikan integritas data dengan mencegah duplikasi dan memastikan bahwa setiap baris dapat memiliki entitas yang jelas.

❓ Mengapa pada kolom no_telp yang menggunakan tipe data CHAR bukan VARCHAR?



Karena nomor telepon biasanya memiliki panjang karakter yang tetap, sehingga menggunakan tipe data CHAR yang tetap panjang lebih efisien daripada VARCHAR yang panjangnya bervariasi.

❓ Mengapa Hanya kolom no_telp yang menggunakan constraint "UNIQUE"?



Karena nomor telepon harus unik untuk setiap entitas dalam basis data, memastikan tidak ada duplikasi dengan menggunakan constraint "UNIQUE" pada kolom no_telp adalah penting.

❓ Mengapa kolom no_telp tidak memakai constraint "NOT NULL", sementara kolom lainnya Menggunakan constraint tersebut?



Karena nomor telepon mungkin tidak selalu tersedia atau tidak wajib diisi dalam setiap entitas, oleh karena itu constraint "NOT NULL" tidak digunakan untuk kolom no_telp.

INSERT

INSERT 1 DATA

STRUKTUR

```
INSERT INTO nama_tabel  
VALUES (nilai1, nilai2, nilai3, nilai4);
```

CONTOH

```
INSERT INTO pelanggan VALUES(1,"Farel","Alfahrezi",'083856721479');
```

HASIL

```
MariaDB [rental_farel]> SELECT * FROM pelanggan;  
+-----+-----+-----+-----+  
| id_pelanggan | nama_depan | nama_belakang | no_telp      |  
+-----+-----+-----+-----+  
|             1 | Farel      | Alfahrezi     | 083856721479 |  
+-----+-----+-----+-----+  
1 row in set (0.001 sec)  
  
MariaDB [rental_farel]> INSERT INTO pelanggan VALUES (2,"Taufik","Muh_Taufik",  
", '085678921348');  
Query OK, 1 row affected (0.013 sec)  
  
MariaDB [rental_farel]> SELECT * FROM pelanggan;  
+-----+-----+-----+-----+  
| id_pelanggan | nama_depan | nama_belakang | no_telp      |  
+-----+-----+-----+-----+  
|             1 | Farel      | Alfahrezi     | 083856721479 |  
|             2 | Taufik     | Muh_Taufik    | 085678921348 |  
+-----+-----+-----+-----+  
2 rows in set (0.001 sec)
```

ANALISIS

`INSERT INTO pelanggan` : Ini menunjukkan bahwa kita ingin menyisipkan data ke dalam tabel bernama "pelanggan".

`VALUES` : Ini menunjukkan bahwa nilai-nilai yang ingin dimasukkan adalah sebagai berikut.

`(1,"Farel","Alfahrezi",'083856721479')` : Ini adalah nilai-nilai yang ingin dimasukkan ke dalam tabel. Urutannya sesuai dengan urutan kolom-kolom dalam tabel.

KESIMPULAN

Perintah ini menyisipkan satu baris data baru ke dalam tabel "pelanggan" dengan nilai-nilai yang spesifik untuk setiap kolom.

Kesimpulan dari kode tersebut adalah sebuah perintah SQL untuk menyisipkan data ke dalam tabel pelanggan. Data yang dimasukkan mencakup ID pelanggan (1), nama depan ("Farel"), nama belakang ("Alfahrezi"), dan nomor telepon ('083856721479').

INSERT >1 DATA

STRUKTUR

```
INSERT INTO nama_tabel  
VALUES (nilai1, nilai2, nilai3, nilai4), (nilai1, nilai2, nilai3, nilai4);
```

CONTOH

```
INSERT INTO pelanggan VALUES (4,"zhafran","Muh_zhafran",'085222666206'),  
(5,"ahsan","ahsan_putar",'088777222872');
```

HASIL

```
MariaDB [rental_farel]> INSERT INTO pelanggan VALUES (4,"zhafran","Muh_zhafran",'085222666206'), (5,"ahsan","ahsan_putar",'088777222872');  
Query OK, 2 rows affected (0.003 sec)  
Records: 2 Duplicates: 0 Warnings: 0
```

```
MariaDB [rental_farel]> SELECT * FROM pelanggan;  
+-----+-----+-----+-----+  
| id_pelanggan | nama_depan | nama_belakang | no_telp |  
+-----+-----+-----+-----+  
| 1 | Farel | Alfahrezi | 083856721479 |  
| 2 | Taufik | Muh_Taufik | 085678921348 |  
| 3 | Fatur | Muh_fathur | 088765432179 |  
| 4 | zhafran | Muh_zhafran | 085222666206 |  
| 5 | ahsan | ahsan_putar | 088777222872 |  
+-----+-----+-----+-----+  
5 rows in set (0.002 sec)
```

ANALISIS

`INSERT INTO pelanggan` : Menunjukkan bahwa kita ingin menyisipkan data ke dalam tabel bernama "pelanggan".

`VALUES` : Menunjukkan bahwa nilai-nilai yang ingin dimasukkan adalah sebagai berikut.

`(4,"zhafran","Muh_zhafran",'085222666206')` ,

`(5,"ahsan","ahsan_putar",'088777222872')` : Ini adalah beberapa set nilai yang ingin

dimasukkan ke dalam tabel. Setiap set nilai dipisahkan oleh koma.

KESIMPULAN

Jadi, perintah ini akan menyisipkan dua baris data atau lebih dari 1 baris baru ke dalam tabel "pelanggan" sekaligus dengan nilai-nilai yang spesifik untuk setiap kolom dalam setiap baris data.

MENYEBUT KOLOM

STRUKTUR

```
INSERT INTO nama_tabel (kolom1,kolom2) VALUES (nilai1,nilai2);
```

CONTOH

```
INSERT INTO pelanggan (nama_depan,id_pelanggan) VALUES ("adiguna",6);
```

HASIL

```
MariaDB [rental_farel]> INSERT INTO pelanggan (nama_depan,id_pelanggan) VALUES ("adiguna",6);
Query OK, 1 row affected (0.004 sec)
```

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

id_pelanggan	nama_depan	nama_belakang	no_telp
1	Farel	Alfahrezi	083856721479
2	Taufik	Muh_Taufik	085678921348
3	Fatur	Muh_fathur	088765432179
4	zhafran	Muh_zhafran	085222666206
5	ahsan	ahsan_putar	088777222872
6	adiguna	NULL	NULL

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

ANALISIS

`INSERT INTO pelanggan` : Menunjukkan bahwa kita ingin menyisipkan data ke dalam tabel bernama "pelanggan".

`(nama_depan, id_pelanggan)` : Ini adalah daftar kolom-kolom tertentu dalam tabel "pelanggan" di mana kita ingin menyisipkan nilai-nilai.

VALUES ("adiguna", 6) : Ini adalah nilai-nilai yang ingin dimasukkan ke dalam kolom-kolom yang ditentukan. Urutannya harus sesuai dengan urutan kolom yang disebutkan sebelumnya.

KESIMPULAN

Jadi, perintah ini akan menyisipkan satu baris data baru ke dalam tabel "pelanggan", hanya dengan mengisi kolom "nama_depan" dan "id_pelanggan", sedangkan kolom-kolom lainnya akan menggunakan nilai defaultnya atau NULL jika tidak diberikan.

SELECT

SELURUH DATA

STRUKTUR

```
SELECT * FROM nama_tabel;
```

CONTOH

```
SELECT * FROM pelanggan;
```

HASIL

```
MariaDB [rental_farel]> SELECT * FROM pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp      |
+-----+-----+-----+-----+
| 1            | Farel      | Alfahrezi     | 083856721479 |
| 2            | Taufik     | Muh_Taufik    | 085678921348 |
| 3            | Fatur      | Muh_fathur     | 088765432179 |
| 4            | zhafran    | Muh_zhafran    | 085222666206 |
| 5            | ahsan      | ahsan_putar    | 088777222872 |
| 6            | adiguna    | NULL           | NULL          |
+-----+-----+-----+-----+
6 rows in set (0.002 sec)
```

ANALISIS

Kode `SELECT * FROM pelanggan;` digunakan untuk mengambil semua data (seluruh kolom) dari tabel "pelanggan". Ini akan menghasilkan output yang menampilkan semua baris dan kolom yang ada dalam tabel tersebut.

KESIMPULAN

Kesimpulan dari `SELECT * FROM pelanggan;` adalah bahwa perintah tersebut bertujuan untuk menampilkan semua data (seluruh kolom) yang ada dalam tabel "pelanggan".

DATA KOLOM TERTENTU

STRUKTUR

```
SELECT nama_kolom1 nama_kolom2 FROM nama_tabel
```

CONTOH

```
SELECT nama_depan FROM pelanggan;
```

```
SELECT id_pelanggan FROM pelanggan;
```

HASIL



```

MariaDB [rental_farel]> SELECT nama_depan FROM pelanggan;
+-----+
| nama_depan |
+-----+
| Farel      |
| Taufik     |
| Fatur      |
| zhafran    |
| ahsan      |
| adiguna    |
+-----+
6 rows in set (0.005 sec)

MariaDB [rental_farel]> SELECT id_pelanggan FROM pelanggan;
+-----+
| id_pelanggan |
+-----+
|             6 |
|             1 |
|             4 |
|             2 |
|             3 |
|             5 |
+-----+
6 rows in set (0.006 sec)

```

ANALISIS

Kode `SELECT nama_depan FROM pelanggan;` digunakan untuk mengambil data dari kolom "nama_depan" dari tabel "pelanggan". Ini akan menghasilkan output yang menampilkan nilai-nilai yang ada dalam kolom "nama_depan" untuk setiap baris dalam tabel tersebut.

KESIMPULAN

Kesimpulan dari `SELECT nama_depan FROM pelanggan;` adalah bahwa perintah tersebut digunakan untuk menampilkan nilai dari kolom "nama_depan" dari tabel "pelanggan".

KLAUSA WHERE

STRUKTUR

```
SELECT nama_kolom FROM nama_tabel WHERE kondisi;
```

Contoh Format Kondisi (nama_kolom, operator, nilai)

nama_kolom : seperti id_pelanggan

Operator : seperti =, >, <, >=, <=, !=, <>

Nilai : seperti 1, 2, 3

CONTOH

```
SELECT nama_depan FROM pelanggan WHERE id_pelanggan=2;
```

```
SELECT nama_belakang FROM pelanggan WHERE id_pelanggan=1;
```

HASIL

```
MariaDB [rental_fare1]> SELECT nama_depan FROM pelanggan WHERE id_pelanggan=
2;
+-----+
| nama_depan |
+-----+
| Taufik     |
+-----+
1 row in set (0.018 sec)

MariaDB [rental_fare1]> SELECT nama_belakang FROM pelanggan WHERE id_pelangg
an=1;
+-----+
| nama_belakang |
+-----+
| Alfahrezi     |
+-----+
1 row in set (0.002 sec)
```

ANALISIS

`SELECT nama_depan` : Ini menunjukkan bahwa kita ingin mengambil nilai dari kolom "nama_depan".

`FROM pelanggan` : Ini menunjukkan bahwa kita ingin mengambil data dari tabel "pelanggan".

`WHERE id_pelanggan=2` : Ini adalah klausa yang digunakan untuk memberikan kriteria pencarian. Dalam hal ini, kita hanya ingin mendapatkan data dari baris yang memiliki nilai "id_pelanggan" sama dengan 2.

KESIMPULAN

Jadi, perintah ini akan mengambil nilai kolom "nama_depan" dari baris di dalam tabel "pelanggan" di mana nilai "id_pelanggan" sama dengan 2.

UPDATE

STRUKTUR

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

CONTOH

```
UPDATE pelanggan SET no_telp="088889999777" WHERE id_pelanggan="2";
```

HASIL

```
MariaDB [rental_farel]> SELECT * FROM pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
| 1 | Farel | Alfahrezi | 083856721479 |
| 2 | Taufik | Muh_Taufik | 088889999777 |
| 3 | Fatur | Muh_fathur | 088765432179 |
| 4 | zhafran | Muh_zhafran | 085222666206 |
| 5 | ahsan | ahsan_putar | 088777222872 |
| 6 | adiguna | NULL | NULL |
+-----+-----+-----+-----+
6 rows in set (0.001 sec)

MariaDB [rental_farel]> UPDATE pelanggan SET no_telp="085343666309" WHERE id_pelanggan="2";
Query OK, 1 row affected (0.085 sec)
Rows matched: 1 Changed: 1 Warnings: 0

MariaDB [rental_farel]> SELECT * FROM pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
| 1 | Farel | Alfahrezi | 083856721479 |
| 2 | Taufik | Muh_Taufik | 085343666309 |
| 3 | Fatur | Muh_fathur | 088765432179 |
| 4 | zhafran | Muh_zhafran | 085222666206 |
| 5 | ahsan | ahsan_putar | 088777222872 |
| 6 | adiguna | NULL | NULL |
+-----+-----+-----+-----+
6 rows in set (0.001 sec)
```

ANALISIS

UPDATE pelanggan : Menunjukkan bahwa kita ingin memperbarui data di dalam tabel "pelanggan".

`SET no_telp="088889999777"` : Menetapkan nilai baru `"088889999777"` ke dalam kolom `"no_telp"`.

`WHERE id_pelanggan="2"` : Ini adalah klausa yang digunakan untuk memberikan kriteria pembaruan. Dalam hal ini, kita hanya ingin memperbarui data pada baris di mana nilai `"id_pelanggan"` sama dengan `2`.

KESIMPULAN

Jadi, perintah ini akan memperbarui nilai kolom `"no_telp"` menjadi `"088889999777"` di dalam tabel `"pelanggan"` di mana nilai `"id_pelanggan"` sama dengan `2`.

HAPUS BARIS DATA

STRUKTUR

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

CONTOH

```
DELETE FROM pelanggan WHERE id_pelanggan="3";
```

HASIL




```

MariaDB [rental_farel]> SELECT * FROM pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
|          1 | Farel      | Alfahrezi     | 083856721479 |
|          2 | Taufik     | Muh_Taufik    | 085343666309 |
|          3 | Fatur      | Muh_fathur    | 088765432179 |
|          4 | zhafran    | Muh_zhafran   | 085222666206 |
|          5 | ahsan      | ahsan_putar   | 088777222872 |
|          6 | adiguna    | NULL          | NULL          |
+-----+-----+-----+-----+
6 rows in set (0.001 sec)

MariaDB [rental_farel]> DELETE FROM pelanggan WHERE id_pelanggan="3";
Query OK, 1 row affected (0.004 sec)

MariaDB [rental_farel]> SELECT * FROM pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
|          1 | Farel      | Alfahrezi     | 083856721479 |
|          2 | Taufik     | Muh_Taufik    | 085343666309 |
|          4 | zhafran    | Muh_zhafran   | 085222666206 |
|          5 | ahsan      | ahsan_putar   | 088777222872 |
|          6 | adiguna    | NULL          | NULL          |
+-----+-----+-----+-----+
5 rows in set (0.001 sec)

```

ANALISIS

`DELETE FROM pelanggan` : Menunjukkan bahwa kita ingin menghapus data dari tabel "pelanggan".

`WHERE id_pelanggan="3"` : Ini adalah klausa yang digunakan untuk memberikan kriteria penghapusan. Dalam hal ini, kita ingin menghapus baris data di mana nilai kolom "id_pelanggan" sama dengan 3.

KESIMPULAN

Jadi, perintah ini akan menghapus baris data dari tabel "pelanggan" di mana nilai "id_pelanggan" sama dengan 3.

HAPUS TABEL

STRUKTUR

```
DROP TABLE nama_tabel;
```

CONTOH

```
DROP TABLE pembuatan;
```

HASIL

```
MariaDB [rental_farel]> SHOW TABLES;
+-----+
| Tables_in_rental_farel |
+-----+
| pelanggan              |
| pembuatan              |
+-----+
2 rows in set (0.003 sec)

MariaDB [rental_farel]> DROP TABLE pembuatan;
Query OK, 0 rows affected (0.009 sec)

MariaDB [rental_farel]> SHOW TABLES;
+-----+
| Tables_in_rental_farel |
+-----+
| pelanggan              |
+-----+
1 row in set (0.002 sec)
```

ANALISIS

Kode `DROP TABLE pembuatan;` digunakan untuk menghapus tabel dengan nama "pembuatan". Perlu diingat bahwa perintah ini bersifat permanen dan akan menghapus seluruh struktur dan data yang terkait dengan tabel tersebut.

KESIMPULAN

Kesimpulan dari `DROP TABLE pembuatan;` adalah bahwa perintah tersebut bertujuan untuk menghapus tabel dengan nama "pembuatan". Ini merupakan tindakan permanen dan akan menghapus seluruh struktur dan data yang terkait dengan tabel tersebut.

Latihan"

STRUKTUR

```
CREATE TABLE nama_tabel( id_mobil int(2) PRIMARY KEY NOT NULL , no_plat
varchar(10) UNIQUE NOT NULL , no_mesin varchar(10) UNIQUE , warna
varchar(10) NOT NULL , pemilik varchar(25) NOT NULL , peminjam varchar(10) ,
harga_rental int(10) );
```

CONTOH

```
CREATE TABLE mobil ( id_mobil int(2) PRIMARY KEY NOT NULL , no_plat
varchar(10) UNIQUE NOT NULL , no_mesin varchar(10) UNIQUE , warna
varchar(10) NOT NULL , pemilik varchar(25) NOT NULL , peminjam varchar(10) ,
harga_rental int(10) );
```

```
INSERT INTO mobil VALUES (1,"DD 2650
XY","ACX3560","HITAM","TAUFIQ","FAREL",50000),
(2,"DD 2440 AX","BCS1120","MERAH","FATUR","AHSAN",100000),
(3,"B 1611 QC","LSQ1112","SILVER","ASEP","CAPO",50000);
```

```
INSERT INTO mobil (id_mobil,no_plat,no_mesin,warna,pemilik,harga_rental)
VALUES (4,"DD 2901 JK","UQL1029","HITAM","JORDAN",'150000'),(5,"DD 2210
LS","CJH1011","HITAM","NAFAN",'100000');
```

HASIL

```
MariaDB [rental_farel]> DESC mobil;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_mobil   | int(2)        | NO   | PRI | NULL    |       |
| no_plat    | varchar(10)   | NO   | UNI | NULL    |       |
| no_mesin   | varchar(10)   | NO   | UNI | NULL    |       |
| warna      | varchar(10)   | NO   |     | NULL    |       |
| pemilik    | varchar(25)   | NO   |     | NULL    |       |
| peminjam   | varchar(10)   | YES  |     | NULL    |       |
| harga_rental | int(10)       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.011 sec)
```

```

MariaDB [rental_farel]> SELECT * FROM mobil;
+-----+-----+-----+-----+-----+-----+-----+
| id_mobil | no_plat   | no_mesin | warna  | pemilik | peminjam | harga_rental |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | DD 2650 XY | ACX3560 | HITAM  | FAREL   | TAUFIQ   | 50000         |
| 2 | DD 2440 AX | BCS1120 | MERAH | FATUR   | AHSAN    | 100000        |
| 3 | B 1611 QC  | LSQ1112 | SILVER | ASEP    | CAPO     | 50000         |
+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.002 sec)

MariaDB [rental_farel]> INSERT INTO mobil (id_mobil,no_plat,no_mesin,warna,pemilik,harga_rental) VALUES (4,"DD 2901 JK","UQL1029","HITAM","JORDAN",'150000'),(5,"DD 2210 LS","CJH1011","HITAM","NAFAN",'100000');
Query OK, 2 rows affected (0.005 sec)
Records: 2  Duplicates: 0  Warnings: 0

MariaDB [rental_farel]> SELECT * FROM mobil;
+-----+-----+-----+-----+-----+-----+-----+
| id_mobil | no_plat   | no_mesin | warna  | pemilik | peminjam | harga_rental |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | DD 2650 XY | ACX3560 | HITAM  | FAREL   | TAUFIQ   | 50000         |
| 2 | DD 2440 AX | BCS1120 | MERAH | FATUR   | AHSAN    | 100000        |
| 3 | B 1611 QC  | LSQ1112 | SILVER | ASEP    | CAPO     | 50000         |
| 4 | DD 2901 JK | UQL1029 | HITAM  | JORDAN  | NULL     | 150000        |
| 5 | DD 2210 LS | CJH1011 | HITAM  | NAFAN   | NULL     | 100000        |
+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.001 sec)

```

TAMPILKAN STRUKTUR TABEL

STRUKTUR

```
DESC nama_tabel
```

CONTOH

```
DESC mobil
```

HASIL

```
MariaDB [rental_farel]> DESC mobil;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_mobil   | int(2)    | NO   | PRI | NULL    |       |
| no_plat    | varchar(10)| NO   | UNI | NULL    |       |
| no_mesin   | varchar(10)| NO   | UNI | NULL    |       |
| warna      | varchar(10)| NO   |     | NULL    |       |
| pemilik    | varchar(25)| NO   |     | NULL    |       |
| peminjam   | varchar(10)| YES  |     | NULL    |       |
| harga_rental | int(10)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.011 sec)
```

TAMPILKAN DAFTAR TABEL

STRUKTUR

```
SHOW TABLES;
```

CONTOH

```
SHOW TABLES;
```

HASIL

```
Query OK, 0 rows affected (0.017 sec)

MariaDB [rental_farel]> SHOW TABLES;
+-----+
| Tables_in_rental_farel |
+-----+
| mobil                  |
| pelanggan              |
+-----+
2 rows in set (0.002 sec)
```