

1. PENGENALAN BASIS DATA

Definisi Basis

Basis adalah Markas ataupun Gudang tempat Bersaran atau Berkumpul

Definisi Data

Data adalah Kumpulan Fakta Dunia Nyata Yang Mewakili Suatu Objek, Seperti Manusia, Barang, dan lain - lain yang direkam ke dalam bentuk angka, huruf, simbol, teks, bunyi, gambar atau juga kombinasinya

Data adalah Sekumpulan Fakta dari Sebuah Objek

Definisi Basis Data

Basis data adalah kumpulan informasi yang disimpan di dalam komputer secara sistematis sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut.

Basis data Sebagai Wadah Data Yang Mempunyai Aturan dan Struktur Tertentu

Kesimpulan

Basis Data adalah Kumpulan Informasi Yang disimpan di dalam Komputer secara Sistematis atau secara Beraturan dan mempunyai struktur tertentu

Peranan Basis Data

Kampus Undipa Saat Ini dilengkapi dengan Sebuah Sistem basis data yang memungkinkan data mahasiswa, Program Studi, Jadwal Kuliah, dan Informasi akademis

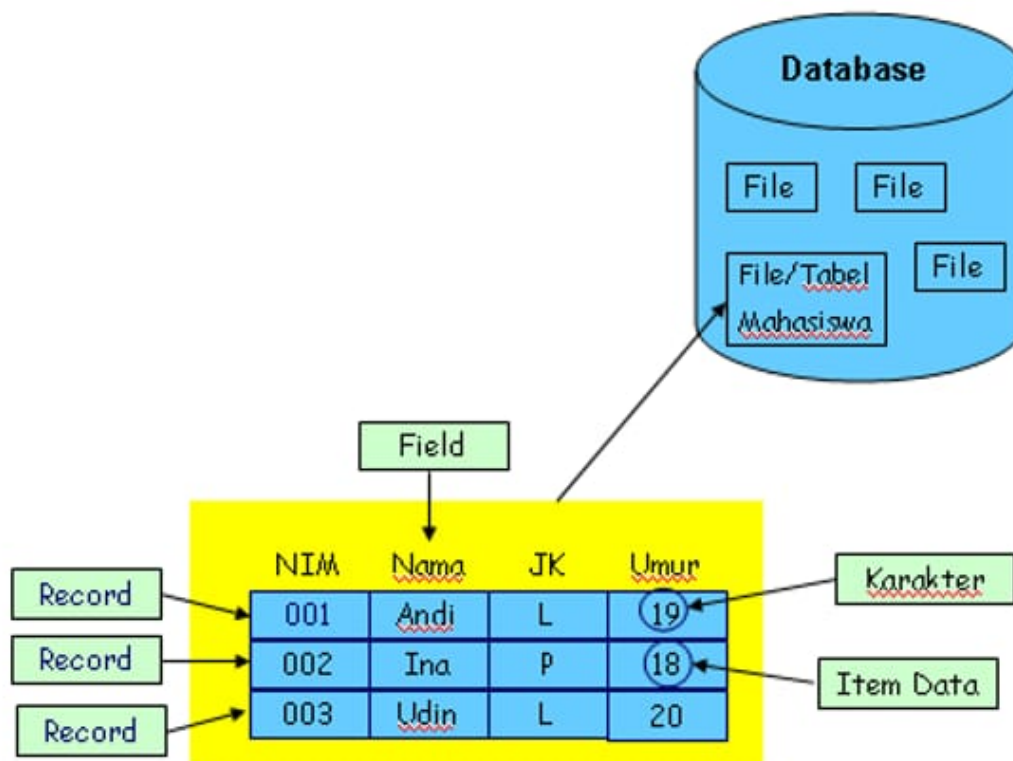
Contoh Data Mahasiswa

Ini Sistem Informasi yang Melibatkan Data Pribadi Mahasiswa Seperti Nama, Alamat, kontak, jenis kelamin, dan Umur. yang dikelola Oleh Staff

Selasa 16-January-2024

2. Tugas Database

Contoh Tabel DataBase



Struktur Tabel Dalam Basis Data

Struktur/Hirarki DataBase adalah struktur organisasi data dalam database yang mengatur hubungan antara entitas atau tabel data. Di dalam hirarki database, data diorganisir dalam

bentuk pohon dengan satu entitas induk atau tabel utama yang memiliki beberapa entitas tabel yang terkait.

Database saya anggap seperti Liga Inggris yang dimana Dia Menampung Club" Seperti MU,MC, yg di dalamnya Mempunyai Pemain Yang berbeda beda Sama Seperti Database dimana Dia Menampung File" Yg berbeda beda di dalamnya

Contohnya:

Ada 4 Club yg saya Ambil 1 saya Uraikan DiMana 4 Club itu ada Manchester United, Manchester city, Tottenham, Aston Villa.

Yang saya Uraikan Manchester City dalam bentuk Tabel.

Tabel DataBasenya :

Nama Pemain	No Punggung	Negara Pemain	Umur Pemain
Ruben Dias	3	Portugal	26 Tahun
Bernardo Silva	20	Portugal	29 Tahun
Kevin De Bruyne	17	Belgium	32 Tahun
Phill Foden	10	Inggris	23 Tahun

Field (kolom) :

Field adalah komponen terkecil dari sebuah record yang menyimpan nilai tunggal atau data spesifik. Field juga dikenal sebagai kolom dalam tabel basis data, dan setiap field dalam tabel mewakili atribut atau jenis data yang berbeda.

Merepresentasikan kolom-kolom dalam tabel, seperti "Nama Pemain," "No Punggung," "Negara Pemain," dan "Umur Pemain."

Record (baris):

Record adalah kumpulan data terkait yang mewakili entitas tunggal atau item dalam basis data.

Merepresentasikan setiap baris atau entri dalam tabel. Misalnya, data untuk Ruben Dias adalah satu record, data untuk Bernardo Silva adalah record lainnya, dan seterusnya.

Item Data:

baris/record yang mewakili entitas atau objek dalam tabel, kolom yang merepresentasikan atribut atau jenis data, dan nilai yang merupakan data yang disimpan dalam sel atau item individual dalam tabel.

Merujuk pada nilai spesifik di dalam setiap kolom. Contohnya, "Ruben Dias" adalah item data untuk kolom "Nama Pemain," "3" untuk "No Punggung," "Portugal" untuk "Negara Pemain," dan "26 Tahun" untuk "Umur Pemain."

Selasa 23-01-2024

3. INSTALASI & QUERY AWAL DATABASE

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 DATABASE S`.

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".

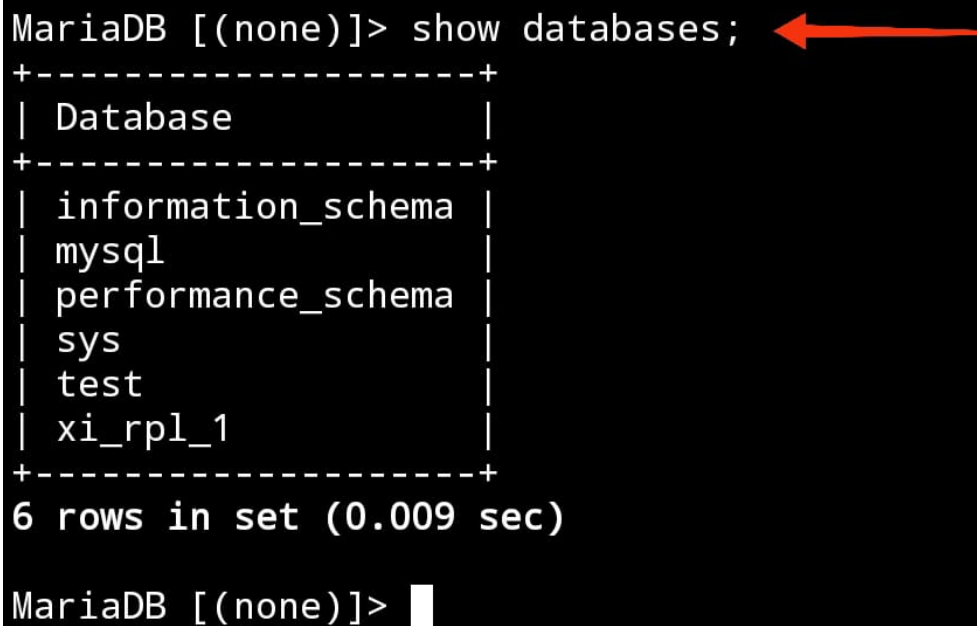
Tampilkan Database

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_rpl_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.

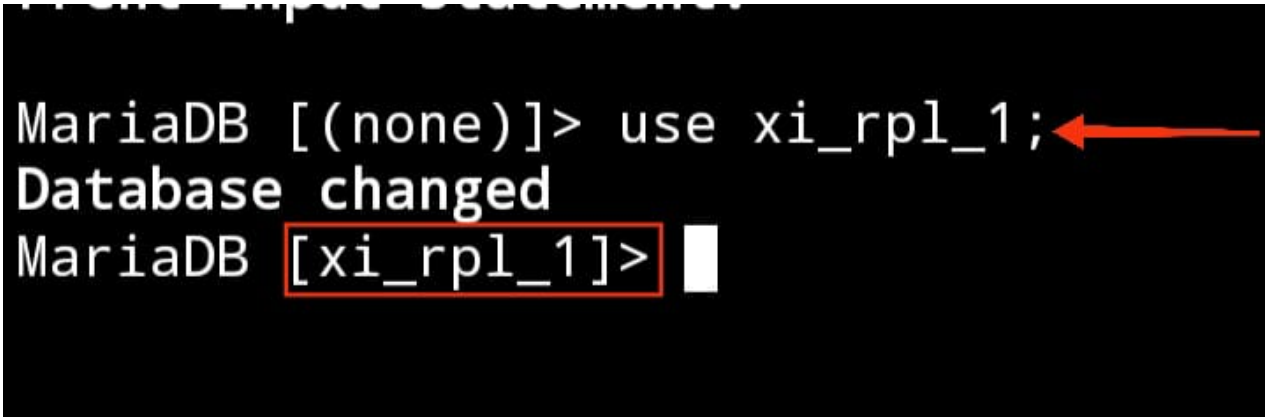
STRUKTUR

```
USE [nama_database];
```

CONTOH

```
USE xi_rpl_1;
```

HASIL



The screenshot shows a terminal window with a black background and white text. The first line shows the prompt 'MariaDB [(none)]>' followed by the command 'use xi_rpl_1;'. A red arrow points to the semicolon at the end of the command. The second line shows the output 'Database changed'. The third line shows the prompt '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

Kode tersebut merupakan perintah untuk membuat tabel baru bernama "pelanggan" dalam database yang aktif. Tabel ini memiliki kolom-kolom seperti "id\_pelanggan" dengan tipe data integer, berperan sebagai kunci utama (PRIMARY KEY) yang tidak boleh kosong (NOT NULL), "nama\_depan" dengan tipe data varchar(25) yang tidak boleh kosong, "nama\_belakang" dengan tipe data varchar(25), dan "no\_telp" dengan tipe data char(12) yang memiliki sifat UNIQUE, yang berarti harus memiliki nilai yang unik di antara semua entri dalam kolom tersebut.

## 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_farel]> show Tables;
+-----+
| Tables_in_rental_farel |
+-----+
| 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 luas 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

Kode tersebut adalah perintah SQL yang bertujuan untuk memasukkan data baru ke dalam tabel "pelanggan". Data yang dimasukkan adalah sebagai berikut:

Data yang dimasukkan melibatkan informasi seperti ID pelanggan (1), nama depan ("Farel"), nama belakang ("Alfahrezi"), dan nomor telepon ('083856721479').

Perintah INSERT INTO digunakan untuk menambahkan data ke dalam tabel yang telah ditentukan. Sintaks VALUES digunakan untuk menentukan nilai-nilai yang akan dimasukkan

ke dalam tabel, sesuai dengan urutan kolom-kolom yang telah didefinisikan dalam tabel tersebut.

## KESIMPULAN

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

Kode tersebut merupakan perintah SQL untuk menyisipkan beberapa baris data sekaligus ke dalam tabel "pelanggan". Data yang dimasukkan mencakup dua baris dengan kolom-kolom yang sama seperti yang telah disebutkan sebelumnya: ID pelanggan, nama depan, nama belakang, dan nomor telepon. Baris pertama memiliki nilai ID pelanggan 4, nama depan "zhafran", nama belakang "Muh\_zhafran", dan nomor telepon '085222666206'. Baris kedua memiliki nilai ID pelanggan 5, nama depan "ahsan", nama belakang "ahsan\_putar", dan nomor telepon '088777222872'.

## KESIMPULAN

Kesimpulan dari kode tersebut adalah perintah SQL untuk menyisipkan dua baris data sekaligus ke dalam tabel "pelanggan". Data tersebut mencakup informasi seperti ID pelanggan, nama depan, nama belakang, dan nomor telepon untuk dua pelanggan dengan ID 4 dan 5.

---

## 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

Kode "INSERT INTO pelanggan VALUES(1, 'Farel', 'Alfahrezi', '083856721479');" digunakan untuk menambahkan baris data baru ke dalam tabel "pelanggan". Nilai yang ditentukan untuk setiap kolom adalah, secara berurutan, 1 untuk kolom "id\_pelanggan", 'Farel' untuk kolom "nama\_depan", 'Alfahrezi' untuk kolom "nama\_belakang", dan '083856721479' untuk kolom "no\_telp".

## KESIMPULAN

Kesimpulan dari "INSERT INTO pelanggan VALUES(1, 'Farel', 'Alfahrezi', '083856721479');" adalah bahwa perintah tersebut digunakan untuk menyisipkan data baru ke dalam tabel "pelanggan" dengan nilai tertentu untuk setiap kolom. Dalam contoh ini, data baru tersebut mencakup ID pelanggan 1, nama depan "Farel", nama belakang "Alfahrezi", dan nomor telepon "083856721479".

## 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_pelanggan=1;
+-----+
| nama_belakang |
+-----+
| Alfahrezi |
+-----+
1 row in set (0.002 sec)
```

## ANALISIS

Kode "SELECT nama\_depan FROM pelanggan WHERE id\_pelanggan=2;" digunakan untuk mengambil nilai dari kolom "nama\_depan" dari tabel "pelanggan" hanya untuk baris yang memiliki nilai "id\_pelanggan" sama dengan 2. Ini memfilter data dan hanya menampilkan hasil yang memenuhi kondisi tertentu.

## KESIMPULAN

Kesimpulan dari "SELECT nama\_depan FROM pelanggan WHERE id\_pelanggan=2;" adalah perintah tersebut bertujuan untuk menampilkan nilai dari kolom "nama\_depan" hanya untuk baris di tabel "pelanggan" yang memiliki 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

Kode "UPDATE pelanggan SET no\_telp='088889999777' WHERE id\_pelanggan='2';" digunakan untuk memperbarui nilai kolom "no\_telp" pada tabel "pelanggan" dengan nilai baru "088889999777" hanya untuk baris yang memiliki nilai "id\_pelanggan" sama dengan 2. Ini adalah perintah untuk mengubah data yang sudah ada dalam tabel.

## KESIMPULAN

Kesimpulan dari "UPDATE pelanggan SET no\_telp='088889999777' WHERE id\_pelanggan='2';" adalah bahwa perintah tersebut digunakan untuk memperbarui nilai kolom "no\_telp" pada tabel "pelanggan" dengan nilai baru "088889999777" hanya untuk baris yang memiliki 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_fare1]> 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_fare1]> DELETE FROM pelanggan WHERE id_pelanggan="3";
Query OK, 1 row affected (0.004 sec)
```

```
MariaDB [rental_fare1]> 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

Kode "DELETE FROM pelanggan WHERE id\_pelanggan='3';" digunakan untuk menghapus baris dari tabel "pelanggan" di mana nilai kolom "id\_pelanggan" sama dengan 3. Ini adalah perintah untuk menghapus data tertentu dari tabel berdasarkan kondisi yang diberikan. Dalam contoh ini, baris dengan "id\_pelanggan" 3 akan dihapus.

## KESIMPULAN

Kesimpulan dari "DELETE FROM pelanggan WHERE id\_pelanggan='3';" adalah bahwa perintah tersebut digunakan untuk menghapus baris dari tabel "pelanggan" di mana nilai kolom "id\_pelanggan" sama dengan 3. Ini adalah perintah untuk menghapus data tertentu dari tabel berdasarkan kondisi yang diberikan. Dalam contoh ini, baris dengan "id\_pelanggan" 3 akan dihapus.

---

## 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",500000),
(2,"DD 2440 AX","BCS1120","MERAH","FATUR","AHSAN",1000000),
(3,"B 1611 QC","LSQ1112","SILVER","ASEP","CAPO",500000);
```

```
INSERT INTO mobil (id_mobil,no_plat,no_mesin,warna,pemilik,harga_rental)
VALUES (4,"DD 2901 JK","UQL1029","HITAM","JORDAN",'1500000'),(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\_fare1]> SHOW TABLES;

```
+-----+
| Tables_in_rental_fare1 |
+-----+
| mobil |
| pelanggan |
+-----+
2 rows in set (0.002 sec)
```