

# Instalasy & query awal Database

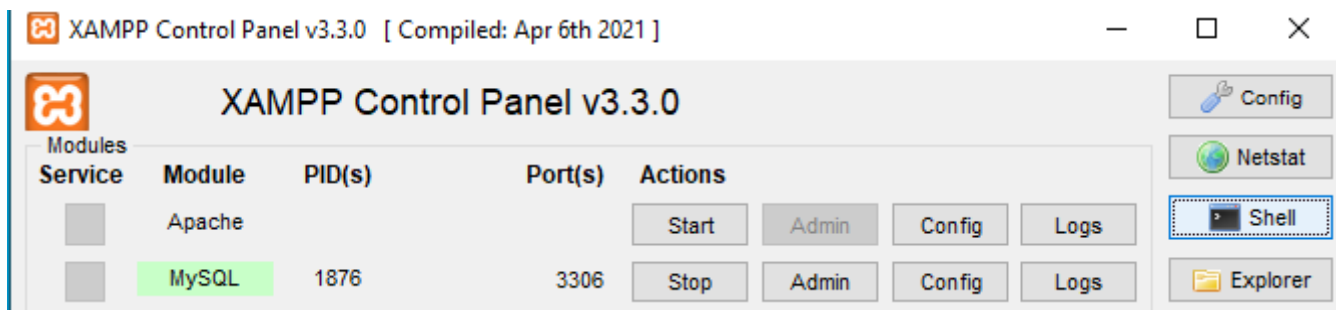
## instalasi mysql

### Menggunakan XAMPP

1. Buka Apk XAMPP
2. Kemudian klik start pada mysql seperti dibawah ini



3. Lalu klik shell pada bagian kanan seperti dibawah ini



4. Sesudah itu ketik mysql -u root -p seperti di bawah ini

```
Setting environment for using XAMPP for Windows.  
ASUS A416M@LAPTOP-U21EUGGU c:\xampp  
# mysql -u root -p  
Enter password:  
Welcome to the MariaDB monitor.  Commands end with ; or \g.  
Your MariaDB connection id is 8  
Server version: 10.4.32-MariaDB mariadb.org binary distribution  
  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
MariaDB [(none)]>
```

5. Selesai

## Refrensi Video dari Youtube

<https://youtu.be/SWs0j2LzcD4?si=1PEEARq-F55bCQbX>



## Penggunaan awal mysql

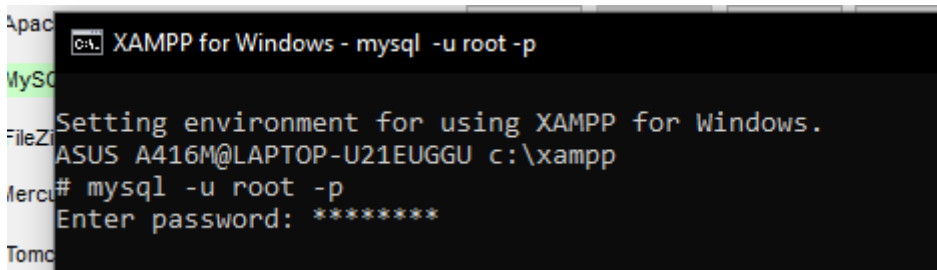
### Query

Perintah `mysql -u root -p` digunakan untuk masuk ke server MySQL dengan mengidentifikasi pengguna "root" dan meminta pengguna untuk memasukkan kata sandi. Mari bahas komponen-komponen perintah ini:

- `mysql`: Memulai klien command-line MySQL.
- `-u root`: Menentukan pengguna (user) yang ingin digunakan untuk login, dalam hal ini "root".
- `-p`: Menyiratkan bahwa sistem akan meminta kata sandi setelah menekan Enter.

Jadi, secara keseluruhan, perintah ini berfungsi untuk mengakses server MySQL sebagai pengguna "root" dan memastikan bahwa hanya pengguna yang memiliki kata sandi yang benar yang dapat mengakses dan berinteraksi dengan server MySQL tersebut.

# Hasil



```
XAMPP for Windows - mysql -u root -p
Setting environment for using XAMPP for Windows.
ASUS A416M@LAPTOP-U21EUGGU c:\xampp
# mysql -u root -p
Enter password: *****
```

## Analisis

Teks "mysql -u root -p" merupakan perintah baris perintah (command line) untuk mengakses server MySQL dengan menggunakan akun pengguna "root" dan meminta pengguna untuk memasukkan kata sandi ("-p" menunjukkan bahwa kata sandi akan dimasukkan setelah menekan Enter).

## Kesimpulan

Kesimpulannya, perintah ini bertujuan untuk masuk ke server MySQL sebagai pengguna "root" dengan mengidentifikasi diri melalui kata sandi yang diperlukan. Dalam konteks ini, pengguna harus memasukkan kata sandi setelah mengetik perintah ini untuk mendapatkan akses ke lingkungan MySQL dengan hak akses yang sesuai dengan pengguna "root".

## Database

### Buat Database

#### query

Perintah

```
CREATE nama_DATABASE;
```

nama\_database digunakan untuk membuat basis data baru di server MySQL.

Contoh:

```
`CREATE DATABASE x1_rpl_1;
```

Fungsinya adalah membuat database dengan nama yang disebutkan setelah kata kunci CREATE DATABASE. Setelah dijalankan, database baru akan dibuat dan dapat digunakan untuk menyimpan tabel, data, dan objek basis data lainnya.

# Hasil

```
MariaDB [(none)]> create database x1_rpl_1;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| phpmyadmin |
| test |
| x1_rpl_1 |
+-----+
6 rows in set (0.001 sec)

MariaDB [(none)]> _
```

## Analisis

Teks "Created database" ini menyatakan bahwa sebuah basis data telah dibuat. Dalam konteks pengelolaan basis data, ini mengindikasikan bahwa pengguna atau sistem telah berhasil membuat suatu basis data baru.

```
`CREATE DATABASE x1_rpl_1;
```

CREATE DATABASE adalah query untuk membuat database dan x1\_rpl\_1 adalah nama database nya.

## Kesimpulan

Kesimpulannya, pernyataan ini menunjukkan bahwa proses pembuatan basis data telah berhasil dilakukan. Selanjutnya, pengguna dapat menggunakan basis data tersebut untuk menyimpan dan mengelola data sesuai kebutuhan aplikasi atau proyek yang sedang dijalankan.

## Tampilkan Database

### Query

Perintah

```
SHOW DATABASES;
```

digunakan pada MySQL untuk menampilkan daftar semua basis data yang ada di server.

## Hasil

```
ASUS A416M@LAPTOP-U21EUGGU c:\xampp
# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 10
Server version: 10.4.32-MariaDB mariadb.org binary distribution

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

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

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| phpmyadmin |
| test |
+-----+
5 rows in set (0.002 sec)

MariaDB [(none)]> _
```

## Analisis

Teks " `SHOW DATABASES;` " merupakan perintah SQL yang digunakan untuk menampilkan daftar semua basis data yang ada di server MySQL.

Fungsinya adalah memberikan informasi mengenai basis data yang tersedia di server MySQL. Ketika perintah ini dijalankan, server akan memberikan daftar nama basis data yang dapat diakses. Ini berguna untuk mengetahui basis data mana yang sudah ada dan tersedia untuk digunakan.

## Kesimpulan

Kesimpulannya, perintah ini digunakan untuk melihat semua basis data yang telah dibuat di server MySQL. Ketika perintah ini dijalankan, server akan merespons dengan menampilkan nama-nama basis data yang tersedia. Hal ini memudahkan pengguna untuk mengetahui basis data apa saja yang sudah ada dalam lingkungan MySQL yang sedang diakses.

## Hapus Database

### Query

Perintah

```
DROP nama_DATABASE
```

digunakan pada MySQL untuk menghapus sebuah basis data beserta seluruh tabel, indeks, dan objek basis data lainnya yang terkait di dalamnya.

Contoh:

```
DROP x1_rpl_1
```

## Hasil

```
MariaDB [(none)]> drop database x1_rpl_1;
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| phpmyadmin     |
| test           |
+-----+
5 rows in set (0.001 sec)

MariaDB [(none)]> _
```

## Analisis

Teks "DROP DATABASE" adalah perintah SQL yang digunakan untuk menghapus sebuah basis data beserta seluruh tabel dan data yang terkait di dalamnya. Fungsinya adalah untuk menghapus secara permanen sebuah basis data dari server MySQL. Harap berhati-hati saat menggunakan perintah ini, karena setelah dieksekusi, semua data yang terkait dengan basis data tersebut akan dihapus dan tidak dapat dikembalikan. Pastikan bahwa Anda yakin ingin menghapus basis data tersebut sebelum menjalankan perintah DROP DATABASE.

```
DROP x1_rpl_1
```

DROP adalah query untuk menghapus database dan x1\_rpl\_1 adalah database yang akan dihapus

## Kesimpulan

Kesimpulannya, perintah ini digunakan dengan hati-hati karena akan menghapus semua informasi yang terkandung dalam basis data yang ditentukan. Pengguna perlu memastikan

bahwa mereka benar-benar ingin menghapus basis data tersebut sebelum menjalankan perintah ini, karena data yang dihapus tidak dapat dipulihkan.

## Gunakan Database

### Query

Perintah

```
USE nama_DATABASE;
```

digunakan pada MySQL untuk beralih ke basis data tertentu, sehingga semua operasi yang dilakukan setelahnya akan berlaku pada basis data tersebut.

Contoh:

```
USE farhan_xi;
```

Fungsinya adalah mengubah basis data aktif yang digunakan dalam sesi koneksi MySQL. Setelah menjalankan perintah USE DATABASE, semua operasi seperti SELECT, INSERT, UPDATE, atau DELETE akan diterapkan pada tabel dalam basis data yang telah dipilih. Ini memungkinkan pengguna untuk fokus pada manipulasi data di dalam basis data tertentu tanpa harus menyertakan nama basis data secara eksplisit setiap saat.

### Hasil

```
MariaDB [(none)]> create database farhan_xi;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| farhan_xi |
| information_schema |
| mysql |
| performance_schema |
| phpmyadmin |
| test |
+-----+
6 rows in set (0.001 sec)

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

### Analisis

Teks "USE nama\_DATABASE" adalah perintah SQL yang digunakan untuk beralih ke basis data tertentu. Basis data yang ditentukan akan menjadi basis data aktif untuk sesi koneksi saat itu.

```
USE farhan_xi;
```

USE adalah query untuk menggunakan database dan farhan\_xi adalah database yang akan digunakan

## Kesimpulan

Kesimpulannya, perintah ini memungkinkan pengguna untuk berpindah antara basis data yang ada dalam server MySQL. Setelah menjalankan perintah "USE DATABASE", semua operasi database selanjutnya dalam sesi tersebut akan diterapkan pada basis data yang telah dipilih, hingga pengguna beralih ke basis data lain atau menutup sesi koneksi.

## Tipe Data

### Angka

Beberapa tipe data angka yang umum digunakan pada MySQL antara lain:

- **DOUBLE:** Mirip dengan **FLOAT**, tetapi dapat menyimpan nilai dengan presisi ganda.
- **DECIMAL:** Cocok untuk nilai desimal yang membutuhkan presisi tetap.
- **Integer (Bilangan Bulat):** Digunakan untuk menyimpan bilangan bulat tanpa desimal, seperti 1, -5, atau 100.
- **Float (Bilangan Desimal):** Untuk menyimpan angka dengan desimal, seperti 3.14 atau -0.5.
- **Long (Bilangan Bulat Panjang):** Digunakan untuk menyimpan bilangan bulat yang sangat besar.
- **Complex (Bilangan Kompleks):** Untuk menyimpan angka kompleks yang terdiri dari bagian real dan imajiner.

Dalam contoh ini, **INTEGER**, **FLOAT**, **DOUBLE**, dan **DECIMAL** adalah tipe data angka yang digunakan untuk kolom-kolom tertentu dalam tabel.



```

MariaDB [sekolah_farhan]> create table tabel_guru(
  -> id_guru int(10) primary key not null,
  -> nama_depan varchar(25) not null,
  -> nama_belakang varchar(25) null,
  -> mapel varchar(50) not null,
  -> jabatan varchar(25) null,
  -> usia int(10) not null,
  -> tanggal_lahir varchar(25) not null);
Query OK, 0 rows affected (0.290 sec)

MariaDB [sekolah_farhan]> desc tabel_guru;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_guru        | int(10)       | NO   | PRI | NULL    |       |
| nama_depan     | varchar(25)   | NO   |     | NULL    |       |
| nama_belakang  | varchar(25)   | YES  |     | NULL    |       |
| mapel          | varchar(50)   | NO   |     | NULL    |       |
| jabatan        | varchar(25)   | YES  |     | NULL    |       |
| usia           | int(10)       | NO   |     | NULL    |       |
| tanggal_lahir | varchar(25)   | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.055 sec)

```

## Teks

Beberapa tipe data teks yang umum digunakan pada MySQL antara lain:

- String: Tipe data yang digunakan untuk menyimpan teks atau karakter. String diwakili oleh serangkaian karakter yang diapit oleh tanda kutip, baik itu tunggal (') atau ganda ("). Contoh: 'hello', "world", atau '123abc'.
- Char: Tipe data yang digunakan untuk menyimpan satu karakter. Char diwakili oleh tanda kutip tunggal ('). Contoh: 'a', 'b', atau '1'.
- CHAR: Menyimpan string dengan panjang tetap.
- VARCHAR: Menyimpan string dengan panjang variabel.
- TEXT: Cocok untuk menyimpan teks panjang, seperti paragraf atau dokumen.

Dalam contoh ini, CHAR, VARCHAR, dan TEXT adalah tipe data teks yang digunakan untuk kolom-kolom tertentu dalam tabel.

```

MariaDB [sekolah_farhan]> insert into tabel_guru
  -> values (1,"Adrianty",null,"Pemrograman Web","Ketua Jurusan",34,"1982-06-29"),
  -> (2,"Ibrahim","Mallobasang","Basis Data","Kepala Sekolah",21,"2000-09-21"),
  -> (3,"Muhammad","Yusuf","Pemodelan Perangkat Lunak",null,28,"1992-12-24"),
  -> (4,"Rusdyansyar",null,"Pemograman Berorientasi Objek","Asisten Kepala Sekolah",25,"1996-01-21");
Query OK, 4 rows affected (0.033 sec)
Records: 4 Duplicates: 0 Warnings: 0

```

## Tanggal

Beberapa tipe data tanggal dan waktu yang umum digunakan pada MySQL antara lain:

- **Date (Tanggal):** Tipe data yang digunakan untuk merepresentasikan tanggal. Biasanya terdiri dari tahun, bulan, dan hari (YYYY-MM-DD). Beberapa bahasa pemrograman memiliki tipe data khusus untuk tanggal, misalnya `datetime.date` di Python.
- **Datetime (Tanggal dan Waktu):** Tipe data yang digunakan untuk merepresentasikan tanggal dan waktu. Selain tahun, bulan, dan hari, tipe data ini juga menyertakan informasi waktu seperti jam, menit, dan detik. Contoh: 2024-01-27 15:30:00. (YYYY-MM-DD HH:MM:SS.)
- **TIME:** Menyimpan waktu dengan format 15:30:00. (HH:MM:SS.)
- **TIMESTAMP:** Mirip dengan `DATETIME`, tetapi dengan rentang waktu yang lebih kecil dan terkait dengan zona waktu server.
- **YEAR:** Menyimpan tahun dengan empat digit. (YYYY)

Dalam contoh ini, `DATE`, `TIME`, `DATETIME`, `TIMESTAMP`, dan `YEAR` adalah tipe data yang digunakan untuk kolom-kolom tertentu dalam tabel untuk menyimpan informasi tanggal dan waktu.

```
MariaDB [sekolah_farhan]> insert into tabel_guru
-> values (5,"Farhan",null,"Pemrograman Web","siswa",17,"2007-03-29");
Query OK, 1 row affected (0.074 sec)
```

```
MariaDB [sekolah_farhan]> select * from tabel_guru;
```

id_guru	nama_depan	nama_belakang	mapel	jabatan	usia	tanggal_lahir
1	Adrianty	NULL	Pemrograman Web	Ketua Jurusan	34	1982-06-29
2	Ibrahim	Mallobasang	Basis Data	Kepala Sekolah	21	2000-09-21
3	Muhammad	Yusuf	Pemodelan Perangkat Lunak	NULL	28	1992-12-24
4	Rusdyansyar	NULL	Pemograman Berorientasi Objek	Asisten Kepala Sekolah	25	1996-01-21
5	Farhan	NULL	Pemrograman Web	siswa	17	2007-03-29

```
5 rows in set (0.027 sec)
```

## Boolean

- tipe data boolean digunakan untuk menyimpan nilai kebenaran, yaitu `True` (benar) atau `False` (salah). Tipe data boolean umumnya digunakan dalam kondisi percabangan dan pengambilan keputusan.

## Tipe Data Pilihan

### Enum

Tipe data `ENUM` (enumeration) dalam basis data adalah tipe data yang digunakan untuk mendefinisikan kumpulan nilai tetap yang dapat diidentifikasi oleh nama. Setiap nilai dalam `ENUM` diberi label dan harus dipilih dari daftar nilai yang telah ditentukan.

### Set

MySQL memiliki tipe data SET yang memungkinkan Anda untuk menyimpan himpunan nilai tetap yang telah ditentukan sebelumnya. Namun, penggunaannya harus dipertimbangkan dengan hati-hati karena dapat menyulitkan pengelolaan dan perubahan struktur data

## Tabel

### Buat Tabel

#### Struktur

```
create table nama_tabel(  
    nama_kolom_1 tipe_data(max karakter) constraint,  
    nama_kolom_2 tipe_data(max karakter) constraint,  
    nama_kolom_3 tipe_data(max karakter),  
    ...);
```

#### Contoh

```
create table biodata(  
    nis_siswa int(8) primary key not null,  
    nama_asli varchar(25) not null,  
    nama_panggilan varchar(25),  
    no_telp char(12) unique);
```

### Tampilkan Struktur Tabel

#### Hasil

```
MariaDB [rental_farhan]> describe biodata;  
+-----+-----+-----+-----+-----+-----+  
| Field      | Type      | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| nis_siswa  | int(8)    | NO   | PRI | NULL    |       |  
| nama_asli  | varchar(25)| NO   |     | NULL    |       |  
| nama_panggilan | varchar(25)| YES  |     | NULL    |       |  
| no_telp    | char(12)  | YES  | UNI | NULL    |       |  
+-----+-----+-----+-----+-----+-----+  
4 rows in set (0.005 sec)  
  
MariaDB [rental_farhan]> _
```

#### Analisis

Program di atas adalah perintah SQL untuk membuat tabel bernama **"biodata"** dengan empat kolom. Berikut analisisnya:



```
MariaDB [rental_farhan]> show tables;
+-----+
| Tables_in_rental_farhan |
+-----+
| biodata                  |
| pelanggan               |
+-----+
2 rows in set (0.001 sec)

MariaDB [rental_farhan]> describe biodata;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| nis_siswa  | int(8)    | NO   | PRI | NULL    |       |
| nama_asli  | varchar(25)| NO   |     | NULL    |       |
| nama_panggilan | varchar(25)| YES  |     | NULL    |       |
| no_telp    | char(12)  | YES  | UNI | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.005 sec)

MariaDB [rental_farhan]>
```

## Q & A

❓ Mengapa hanya kolom `id_pelanggan` yang menggunakan constraint PRIMARY KEY?

= Jika kolom `id_pelanggan` menggunakan tipe data CHAR, mungkin karena panjang ID pelanggan selalu tetap, dan penggunaan CHAR dapat mengoptimalkan penyimpanan dalam situasi tersebut. Jika menggunakan VARCHAR, itu akan menyimpan panjang yang bervariasi dan membutuhkan sedikit lebih banyak ruang.

❓ mengapa pada kolom `no_telepon` yang menggunakan tipe data char bukan varchar?

=Pemilihan antara tipe data CHAR dan VARCHAR untuk kolom `no_telepon` mungkin tergantung pada kebutuhan dan preferensi desain basis data. Jika panjang nomor telepon selalu tetap dan memiliki panjang yang konsisten, menggunakan CHAR bisa menjadi pilihan yang efisien karena menyimpan data dengan panjang tetap.

❓ Mengapa hanya kolom `no_telepon` yang menggunakan constraint UNIQUE?

Pemberian constraint UNIQUE pada kolom `no_telepon` bertujuan untuk memastikan bahwa setiap nomor telepon yang dimasukkan ke dalam basis data adalah unik. Hal ini dapat membantu mencegah duplikasi nomor telepon yang dapat menyebabkan masalah seperti kesulitan dalam mengidentifikasi dan mengelola data.

❓ Mengapa kolom no\_telepon tidak memakai constraint NOT NULL, sementara kolom lainnya menggunakan constraint tersebut?

Pemberian constraint NOT NULL pada suatu kolom menunjukkan bahwa nilai dalam kolom tersebut tidak boleh kosong (NULL).

❓ Perbedaan PK dan UNIQUE?

primary key digunakan untuk mengidentifikasi unik suatu baris dan sering kali menjadi dasar untuk hubungan antar-tabel, sementara unique constraint digunakan untuk memastikan bahwa suatu kolom memiliki nilai yang unik tetapi tidak selalu terkait dengan identifikasi unik baris.

## Insert

### Insert 1 data

#### Struktur

```
insert into nama_table  
values ('nilai1','nilai2','nilai3','nilai4');
```

#### Contoh

```
insert into pelanggan  
values (1,'farhan','m1n','088247291854');
```

#### Hasil

```

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| farhan_xi |
| information_schema |
| mysql |
| performance_schema |
| phpmyadmin |
| rental_farhan |
| test |
+-----+
7 rows in set (0.004 sec)

MariaDB [(none)]> use rental_farhan;
Database changed
MariaDB [rental_farhan]> insert into pelanggan
    -> values (1, farhan , mln, 088247291854);
ERROR 1054 (42S22): Unknown column 'farhan' in 'field list'
MariaDB [rental_farhan]> insert into pelanggan
    -> values ('1','farhan','mln','088247291854');
Query OK, 1 row affected (0.006 sec)

MariaDB [rental_farhan]> select * from pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
| 1 | farhan | mln | 088247291854 |
+-----+-----+-----+-----+
1 row in set (0.001 sec)

MariaDB [rental_farhan]>

```

## Analisis

1. `insert into pelanggan;` :  
`insert into` merupakan sebuah query untuk menambahkan data baru ke dalam sebuah tabel di database `pelanggan` adalah nama tabelnya
2. `values (1,'farhan','mln','088247291854');` :  
`values` digunakan untuk menyediakan nilai nilai yang akan dimasukkan ke dalam sebuah tabel.
  - Angka `1` akan dimasukkan ke dalam kolom pertama
  - String `'farhan'` akan dimasukkan ke dalam kolom kedua.
  - String `'mln'` akan dimasukkan ke dalam kolom ketiga.
  - String `'088247291854'` akan dimasukkan ke dalam kolom keempat.

## Kesimpulan

kesimpulannya adalah perintah ini akan menambahkan satu baris data baru ke dalam tabel "pelanggan" dengan nilai-nilai yang sesuai yang telah ditentukan.

# insert >1 data

## struktur

```
insert into nama_table
values ('nilai1','nilai2','nilai3','nilai4'),
('nilai1','nilai2','nilai3','nilai4'),
('nilai1','nilai2','nilai3','nilai4');
```

## Contoh

```
insert into pelanggan
values (2,'rehan','alfa','087868449445'),
(3,'ardy','rd','0895333405548'),
(4,'ilham','vp','087733345678');
```

## Hasil

```
MariaDB [(none)]> use rental_farhan;
Database changed
MariaDB [rental_farhan]> insert into pelanggan
-> values (2,'rehan','alfa','087868449445'),
-> (3,'ardy','rd','0895333405548'),
-> (4,'ilham','vp','087733345678');
Query OK, 3 rows affected, 1 warning (0.003 sec)
Records: 3 Duplicates: 0 Warnings: 1

MariaDB [rental_farhan]> select * from pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
| 1 | farhan | mln | 088247291854 |
| 2 | rehan | alfa | 087868449445 |
| 3 | ardy | rd | 089533340554 |
| 4 | ilham | vp | 087733345678 |
+-----+-----+-----+-----+
4 rows in set (0.000 sec)

MariaDB [rental_farhan]>
```

## Analisis

1. insert into pelanggan :

insert into merupakan sebuah query untuk menambahkan data baru ke dalam sebuah tabel di database pelanggan adalah nama tabelnya

2. values (2,'rehan','alfa','087868449445'),  
(3,'ardy','rd','0895333405548'),



```
(4,'ilham','vp','087733345678'); :
```

`values` digunakan untuk menyediakan nilai nilai yang akan dimasukkan ke dalam sebuah tabel.

- Angka `2,3,4` adalah baris yang akan dimasukkan ke dalam kolom pertama
- String `'rehan','ardy','ilham'` akan dimasukkan ke dalam kolom kedua dan sesuai dengan barisnya.
- String `'alfa','rd','vp'` akan dimasukkan ke dalam kolom ketiga dan sesuai dengan barisnya.
- String `'087868449445','0895333405548','087733345678'` akan dimasukkan ke dalam kolom keempat dan sesuai dengan barisnya.

## Kesimpulan

Kesimpulannya, perintah ini akan menambahkan tiga baris data baru ke dalam tabel "pelanggan" dengan nilai-nilai yang sesuai untuk setiap baris yang disediakan.

## Menyebut Kolom

### Struktur

```
insert into nama_table  
    (kolom1,kolom2) values (nilai1,nilai2)
```

### Contoh

```
insert into pelanggan  
    (nama_depan,id_pelanggan) values ('radit',5);
```

## Hasil

```
MariaDB [rental_farhan]> select * from pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
| 1 | farhan | mln | 088247291854 |
| 2 | rehan | alfa | 087868449445 |
| 3 | ardy | rd | 089533340554 |
| 4 | ilham | vp | 087733345678 |
| 5 | radit | NULL | NULL |
+-----+-----+-----+-----+
5 rows in set (0.000 sec)

MariaDB [rental_farhan]> _
```

## Analisis

1. `insert into pelanggan;` :

`insert into` merupakan sebuah query untuk menambahkan data baru ke dalam sebuah tabel di database `pelanggan` adalah nama tabelnya

2. `(nama_depan,id_pelanggan) values ('radit',5);` :

- Tabel yang dituju adalah "pelanggan".
- Data yang akan dimasukkan ke dalam tabel adalah nama depan "radit" dan ID pelanggan 5.

`(nama_depan,id_pelanggan)` sebelum klausa `VALUES` . Ini menunjukkan bahwa hanya kolom-kolom yang ditentukan (yaitu `nama_depan` dan `id_pelanggan` ) yang akan menerima nilai, sedangkan kolom-kolom lainnya akan memiliki nilai default (jika ada) atau NULL.

## Kesimpulan

Kesimpulannya, perintah ini akan menambahkan satu baris data baru ke dalam tabel "pelanggan" dengan nilai nama depan "radit" dan ID pelanggan 5, sementara kolom-kolom lainnya mungkin akan memiliki nilai default atau NULL,

## Select

## Seluruh Data

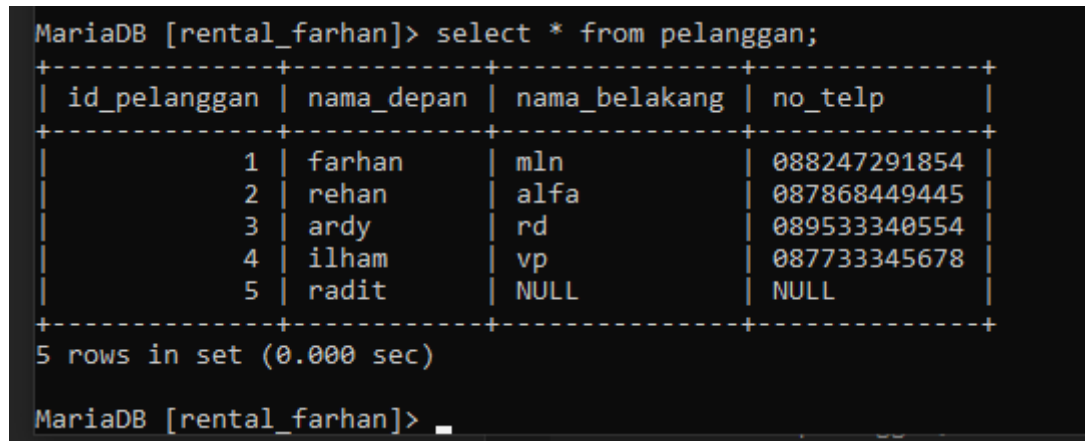
## Struktur

```
select * from nama_tabel;
```

## Contoh

```
select * from pelanggan;
```

## Hasil



The screenshot shows a terminal window with the following content:

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

id_pelanggan	nama_depan	nama_belakang	no_telp
1	farhan	mln	088247291854
2	rehan	alfa	087868449445
3	ardy	rd	089533340554
4	ilham	vp	087733345678
5	radit	NULL	NULL

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

```
MariaDB [rental_farhan]> _
```

## Analisis

```
select * from pelanggan;:
```

Perintah `SELECT * from pelanggan` adalah pernyataan dalam SQL yang digunakan untuk memilih semua kolom dari sebuah tabel.

## Kesimpulan

Kesimpulannya, Perintah `SELECT * FROM pelanggan;` memberikan gambaran keseluruhan tentang data yang tersimpan dalam tabel "pelanggan" pada saat perintah tersebut dijalankan.

## Data kolom tertentu

## Struktur

```
select nama_kolom1,nama_kolom2,nama_kolom_n from nama_table;
```

## Contoh

```
select nama_depan from pelanggan;
```

## Hasil

```
MariaDB [rental_farhan]> select nama_depan from pelanggan;
+-----+
| nama_depan |
+-----+
| farhan     |
| rehan      |
| ardy       |
| ilham      |
| radit      |
+-----+
5 rows in set (0.001 sec)

MariaDB [rental_farhan]>
```

## Analisis

```
select nama_depan from pelanggan;
```

Perintah ini akan mengembalikan nilai dari kolom "nama\_depan" untuk setiap baris data yang ada dalam tabel "pelanggan".

## Kesimpulan

Kesimpulan:

Dengan menggunakan perintah `SELECT nama_depan FROM pelanggan;`, Anda meminta sistem database untuk mengembalikan atau menampilkan hanya kolom "nama\_depan" dari tabel "pelanggan".

## Klausula WHERE

### Struktur

```
select nama_kolom / * from nama_table
where kondisi;
```

### Contoh

```
select * from pelanggan
where id_pelanggan=2;
```

## Hasil

```
MariaDB [rental_farhan]> select nama_depan from pelanggan;
+-----+
| nama_depan |
+-----+
| farhan      |
+-----+
1 row in set (0.001 sec)
```

## Analisis

```
select * from pelanggan where id_pelanggan=2; :
```

Perintah ini akan mengembalikan semua kolom untuk baris data yang memenuhi kondisi di mana nilai kolom "id\_pelanggan" adalah 2.

## Kesimpulan

Kesimpulan:

Dengan menggunakan perintah ini, Anda meminta sistem database untuk mengembalikan atau menampilkan semua informasi yang tersedia untuk pelanggan dengan ID 2 dari tabel "pelanggan"

## Update (Perbarui Data)

### struktur

```
update nama_tabel set nama_kolom="nilai" where kondisi;
```

### contoh

```
update pelanggan set no_telp="085657199460" where id_pelanggan="4";
```

### Hasil

```

MariaDB [rental_farhan]> select * from pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
| 1 | farhan | mln | 088247291854 |
| 2 | rehan | alfa | 087868449445 |
| 3 | ardy | rd | 089533340554 |
| 4 | ilham | vp | 087733345678 |
| 5 | radit | NULL | NULL |
+-----+-----+-----+-----+
5 rows in set (0.007 sec)

MariaDB [rental_farhan]> update pelanggan set no_telp="085657199460" where id="4";
ERROR 1054 (42S22): Unknown column 'id' in 'where clause'
MariaDB [rental_farhan]> update pelanggan set no_telp="085657199460" where id_pelanggan="4";
Query OK, 1 row affected (0.002 sec)
Rows matched: 1 Changed: 1 Warnings: 0

MariaDB [rental_farhan]> select * from pelanggan;
+-----+-----+-----+-----+
| id_pelanggan | nama_depan | nama_belakang | no_telp |
+-----+-----+-----+-----+
| 1 | farhan | mln | 088247291854 |
| 2 | rehan | alfa | 087868449445 |
| 3 | ardy | rd | 089533340554 |
| 4 | ilham | vp | 085657199460 |
| 5 | radit | NULL | NULL |
+-----+-----+-----+-----+
5 rows in set (0.000 sec)

MariaDB [rental_farhan]> _

```

## Analisis

```
update pelanggan set no_telp="085657199460" where id_pelanggan="4"; :
```

- Perintah ini akan mengubah nilai kolom "no\_telp" menjadi "085657199460" untuk baris data yang memenuhi kondisi di mana nilai kolom "id\_pelanggan" adalah 4.
- Perintah ini hanya memengaruhi satu baris data yang memenuhi kondisi tersebut.

## Kesimpulan

Kesimpulan:

Dengan menggunakan perintah UPDATE, Anda dapat memperbarui nilai dari satu atau beberapa kolom untuk baris data tertentu dalam tabel. Dalam hal ini, nomor telepon ("no\_telp") untuk pelanggan dengan ID 4 akan diperbarui menjadi "085657199460"

## Delete (Hapus Baris Data)

### Struktur

```
delete from nama_table where kondisi;
```

## Contoh

```
delete from pelanggan where id_pelanggan="3";
```

## Hasil

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

id_pelanggan	nama_depan	nama_belakang	no_telp
1	farhan	mln	088247291854
2	rehan	alfa	087868449445
3	ardy	rd	089533340554
4	ilham	vp	085657199460
5	radit	NULL	NULL

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

```
MariaDB [rental_farhan]> delete from pelanggan where id_pelanggan="3";  
Query OK, 1 row affected (0.004 sec)
```

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

id_pelanggan	nama_depan	nama_belakang	no_telp
1	farhan	mln	088247291854
2	rehan	alfa	087868449445
4	ilham	vp	085657199460
5	radit	NULL	NULL

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

```
MariaDB [rental_farhan]>
```

## Analisis

```
delete from pelanggan where id_pelanggan="3"; :
```

- Perintah ini akan menghapus satu baris data dari tabel "pelanggan" yang memiliki nilai kolom "id\_pelanggan" sama dengan 3.
- Setelah perintah ini dijalankan, baris data tersebut akan dihapus dari tabel "pelanggan".

## Kesimpulan

Kesimpulan:

Dengan menggunakan perintah DELETE FROM, Anda dapat menghapus baris data yang memenuhi kondisi tertentu dari tabel. Dalam hal ini, baris data yang memiliki nilai "id\_pelanggan" sama dengan 3 akan dihapus dari tabel "pelanggan".

# Hapus Tabel

## struktur

```
drop table nama_tabel;
```

## Contoh

```
drop table biodata;
```

## Hasil

```
MariaDB [rental_farhan]> show tables;
+-----+
| Tables_in_rental_farhan |
+-----+
| biodata                  |
| pelanggan                |
+-----+
2 rows in set (0.001 sec)

MariaDB [rental_farhan]> drop biodata;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual
  line 1
MariaDB [rental_farhan]> drop table biodata;
Query OK, 0 rows affected (0.016 sec)

MariaDB [rental_farhan]> show tables;
+-----+
| Tables_in_rental_farhan |
+-----+
| pelanggan                |
+-----+
1 row in set (0.001 sec)

MariaDB [rental_farhan]>
```

## Analisis

```
drop table biodata;:
```

- Perintah ini akan menghapus seluruh struktur tabel, termasuk semua data dan indeks yang terkait, dari database.
- Setelah perintah ini dijalankan, tabel "biodata" beserta semua informasi yang terkait dengannya akan dihapus dari database



# Kesimpulan

Kesimpulan:

Dengan menggunakan perintah DROP TABLE, Anda dapat menghapus tabel secara permanen dari database.