**FORM PENILAIAN MANDIRI**

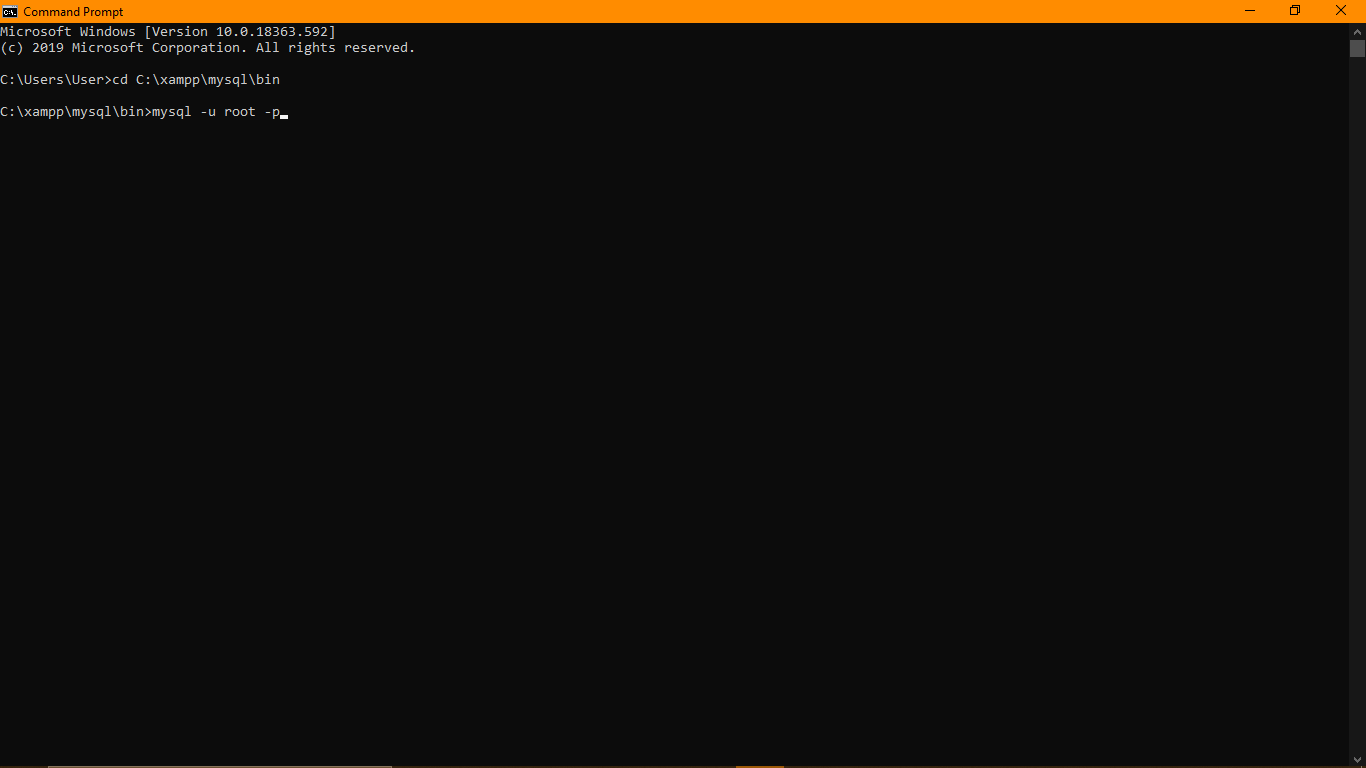
**PEMRROGRAMAN WEB DAN MOBILE**

|  |  |
| --- | --- |
| **Absen** | **Nama** |
| 15 | Marsa Nabil Hawari |
|  |  |

|  |  |
| --- | --- |
| Materi | Nilai |
| Pembelajaran Basis-Data  Video ke 18  PDF – Belajar MYSQL Dari Nol (DDL dan DML)  (Tugas semester 5 #11) | 85 |

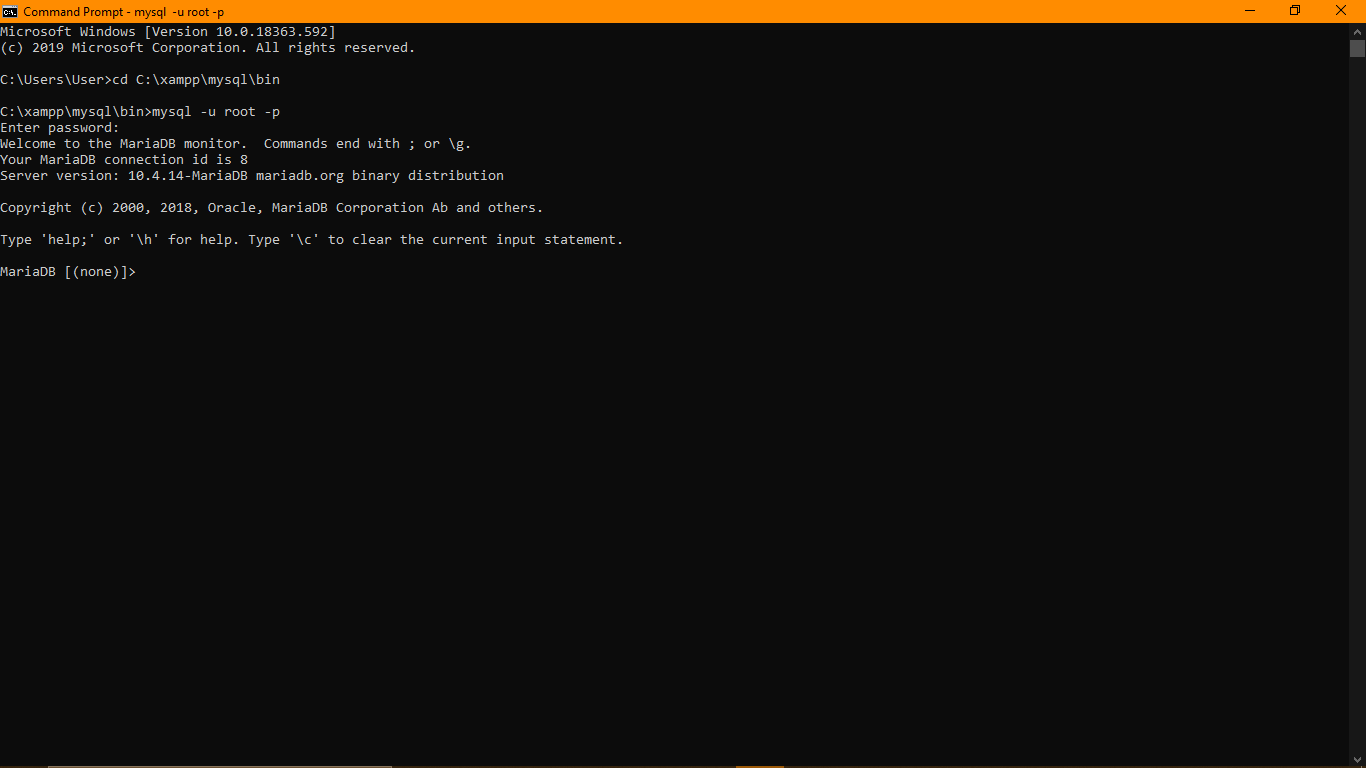
**Saya Sudah Belajar dan Mengerti dan Saya BISA**

1.



Menjalankan Mysql

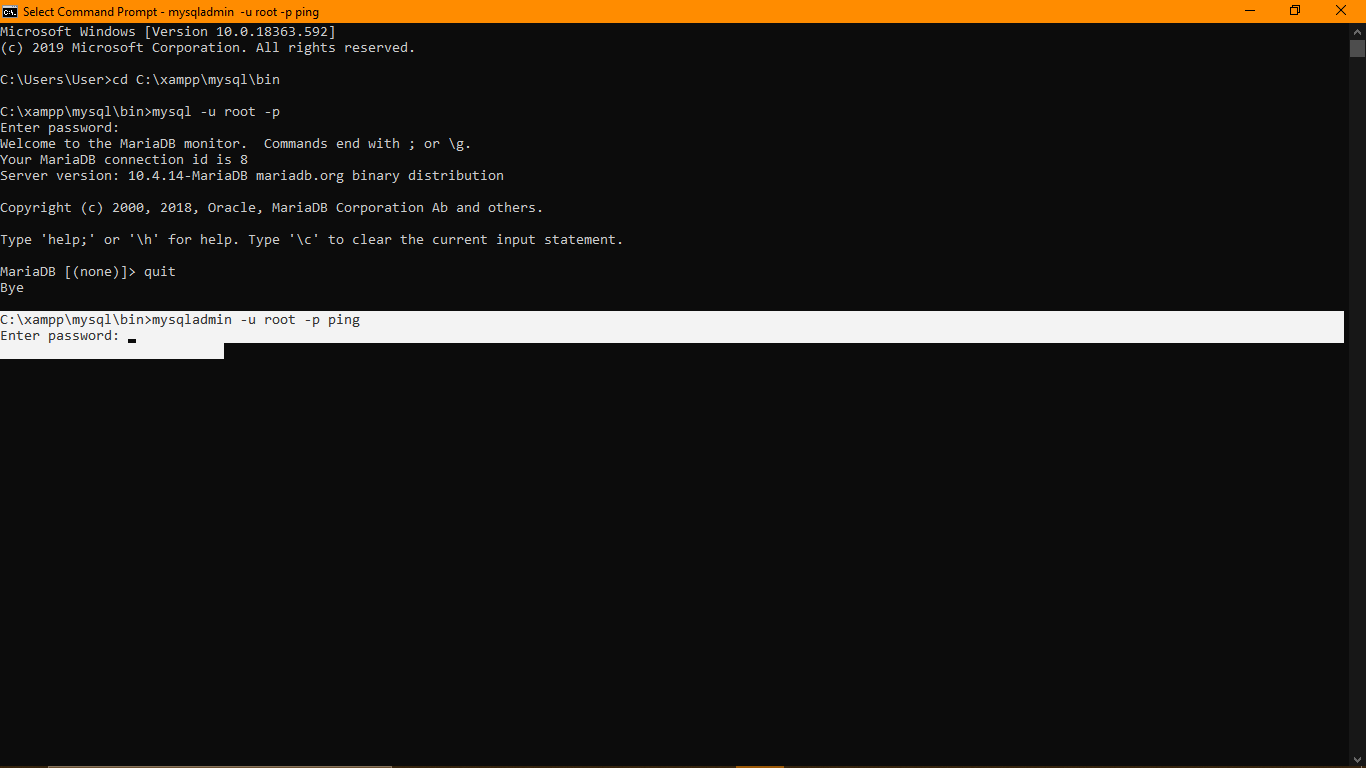
Hasil:



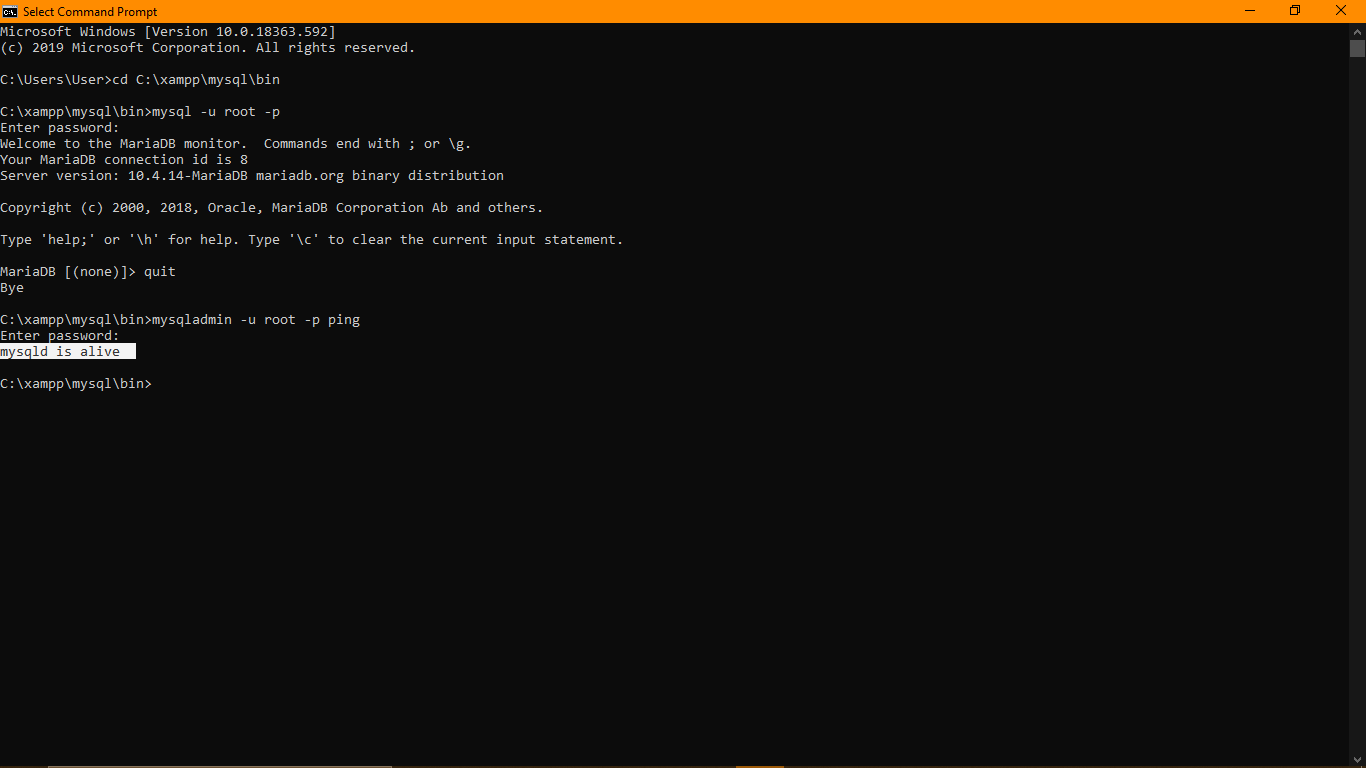
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Untuk keluar dari MySQL gunakan perintah;



Untuk mengecek MySQL berjalan atau tidak, pastikan keluar dari MySQL dulu kemudian gunakan perintah diatas

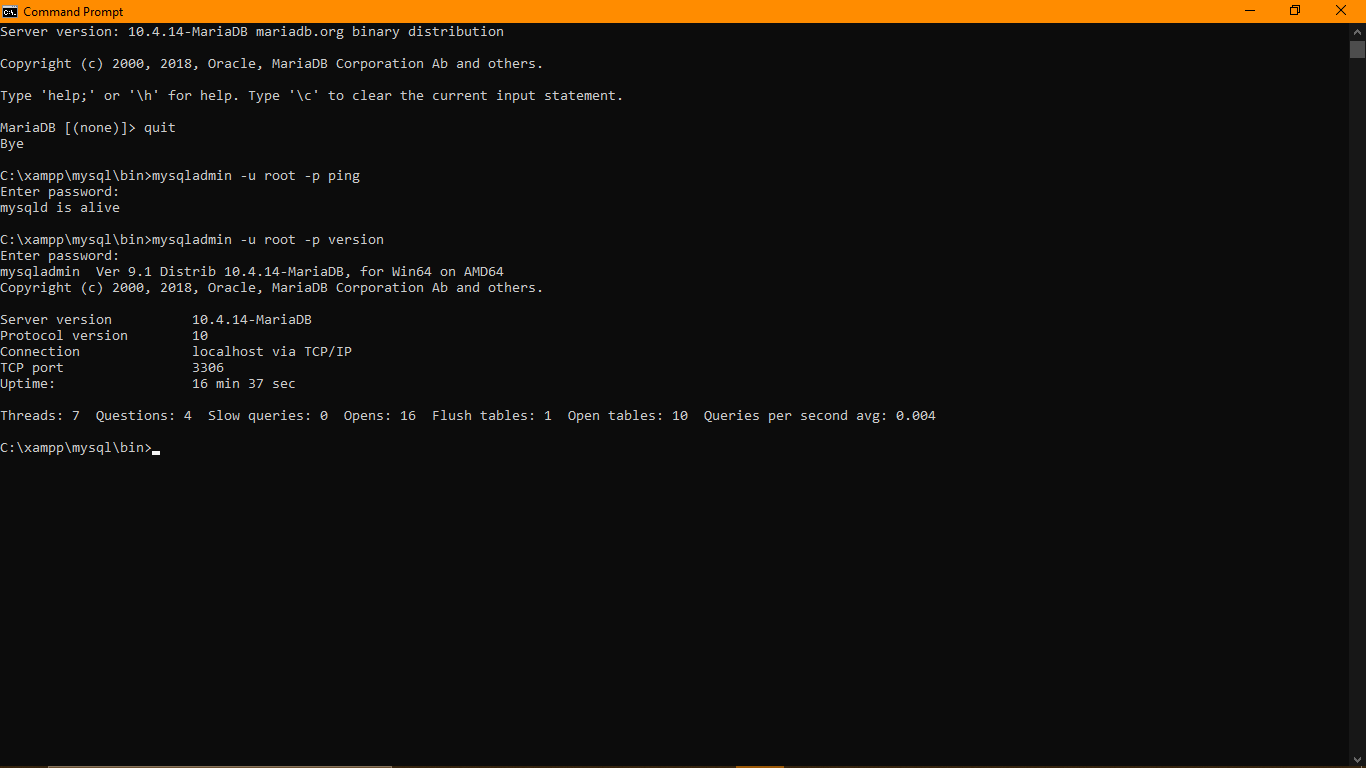


Hasil

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Kita bisa mengecek versi MySQL yang digunakan menggunakan perintah; Pastikan sudah keluar dari MySQL sebelum mengecek versinya.

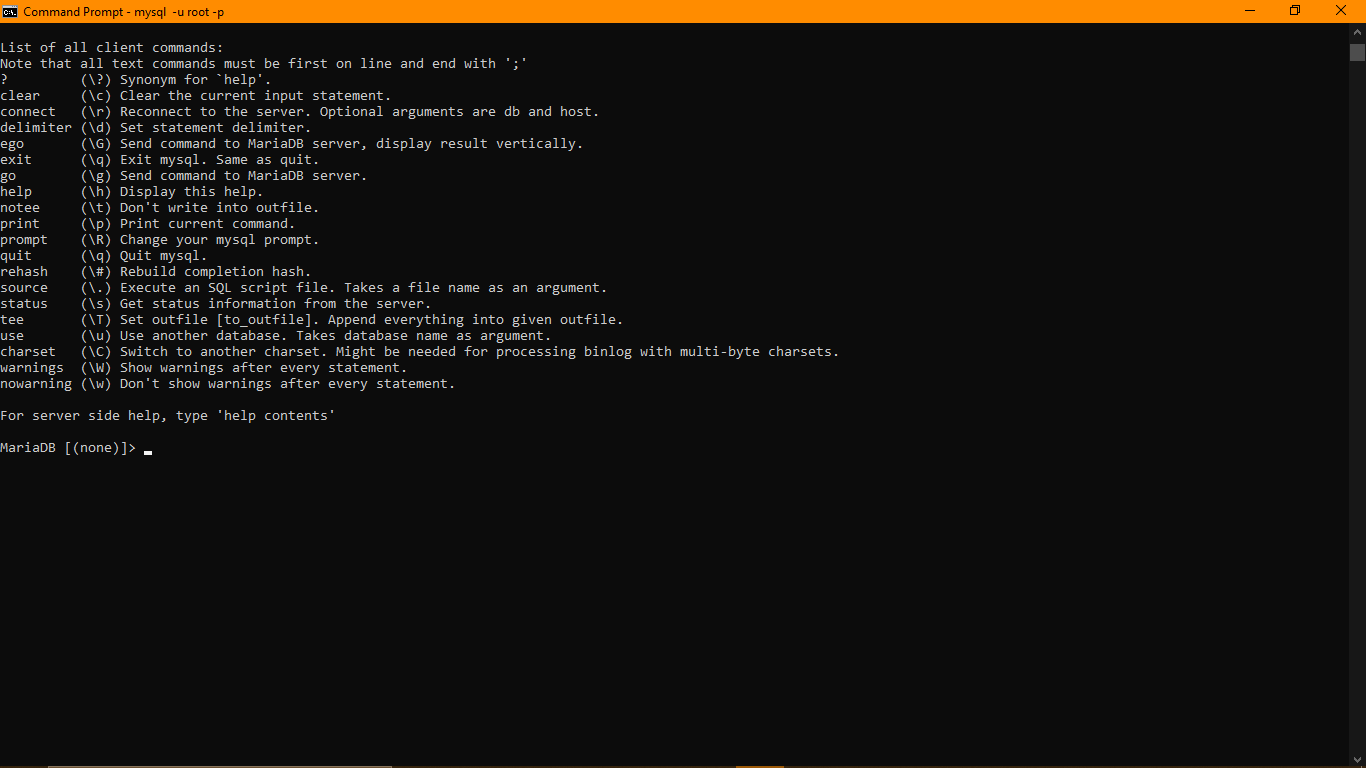


Tampilan hasil pengecekan versi MySQL

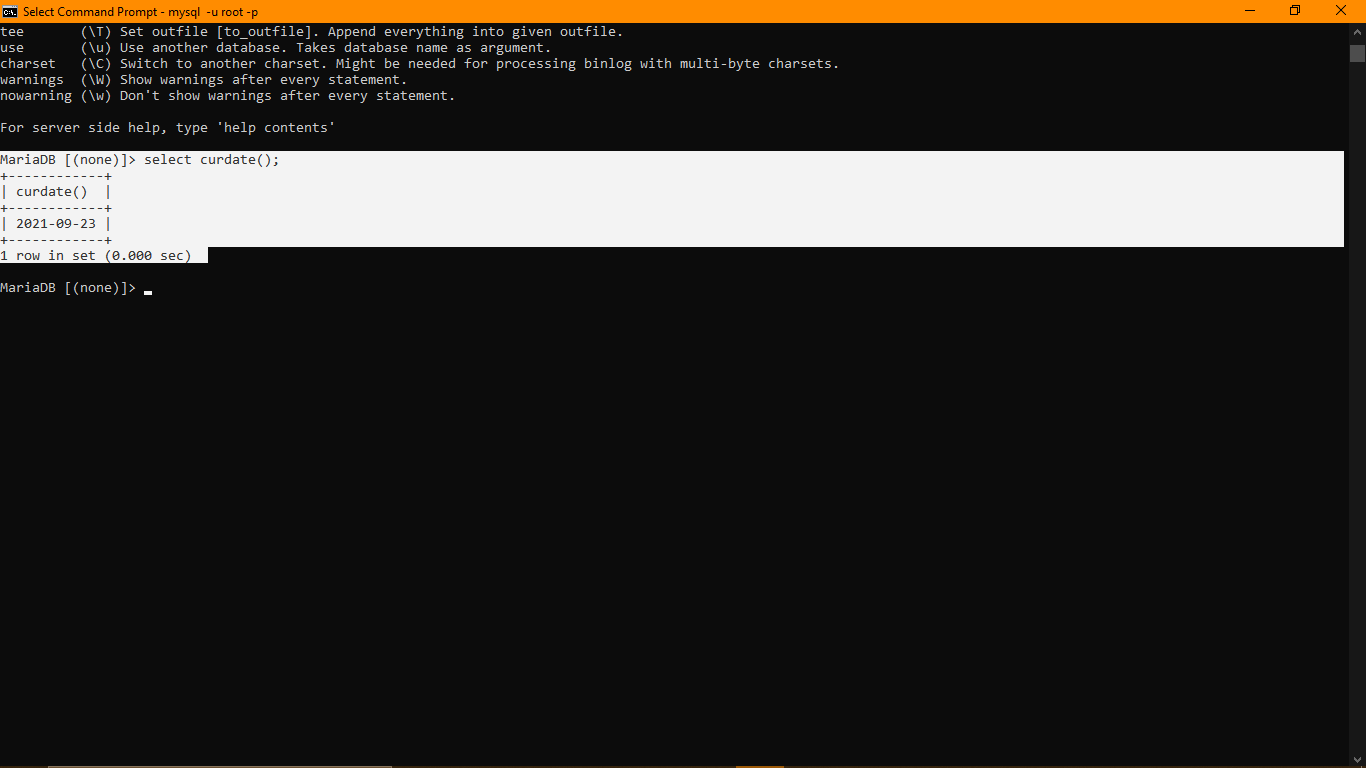




Sangat sulit mengingat semua perintah MySQL, karena itu dibutuhkan bantuan agar kita bisa mengetahui perintah MySQL yang akan digunakan. Untuk melihat bantuan MySQL, anda harus masuk dulu ke MySQL dengan perintah diatas

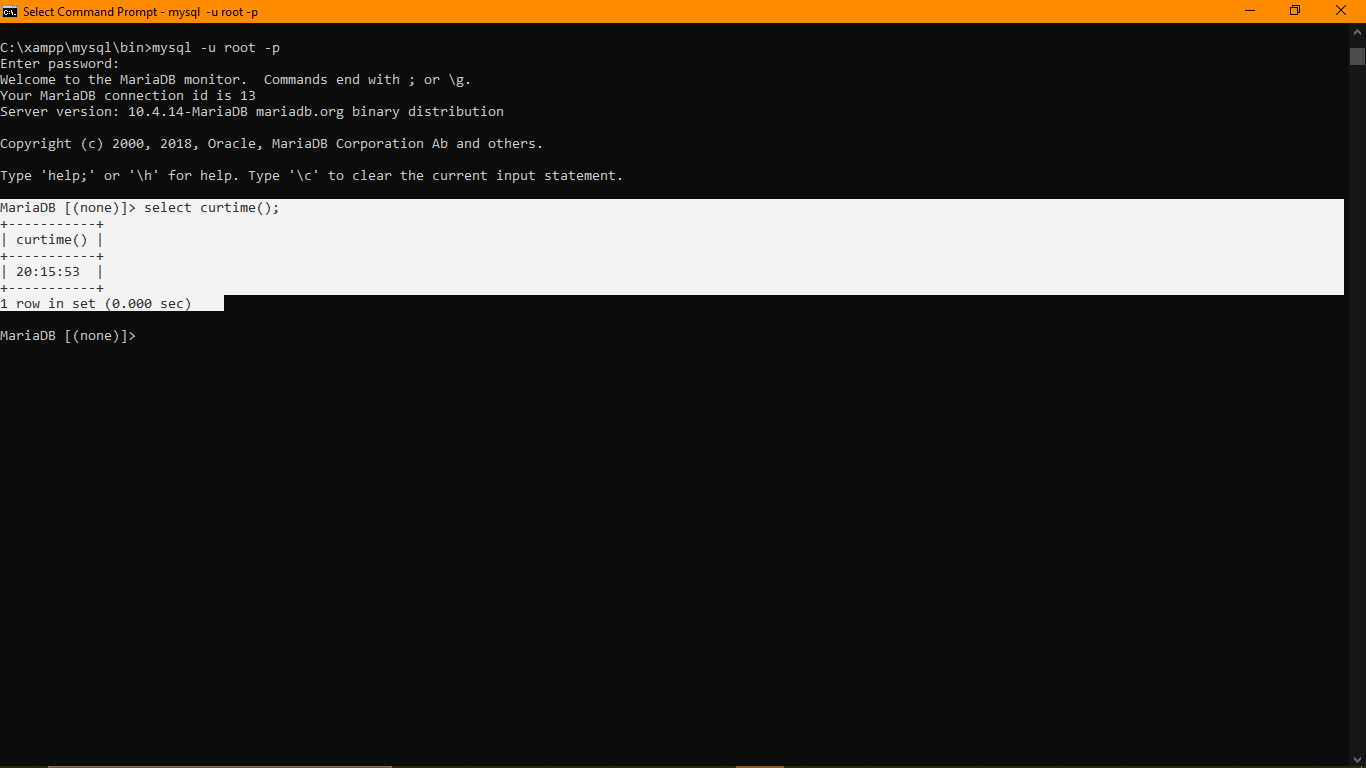


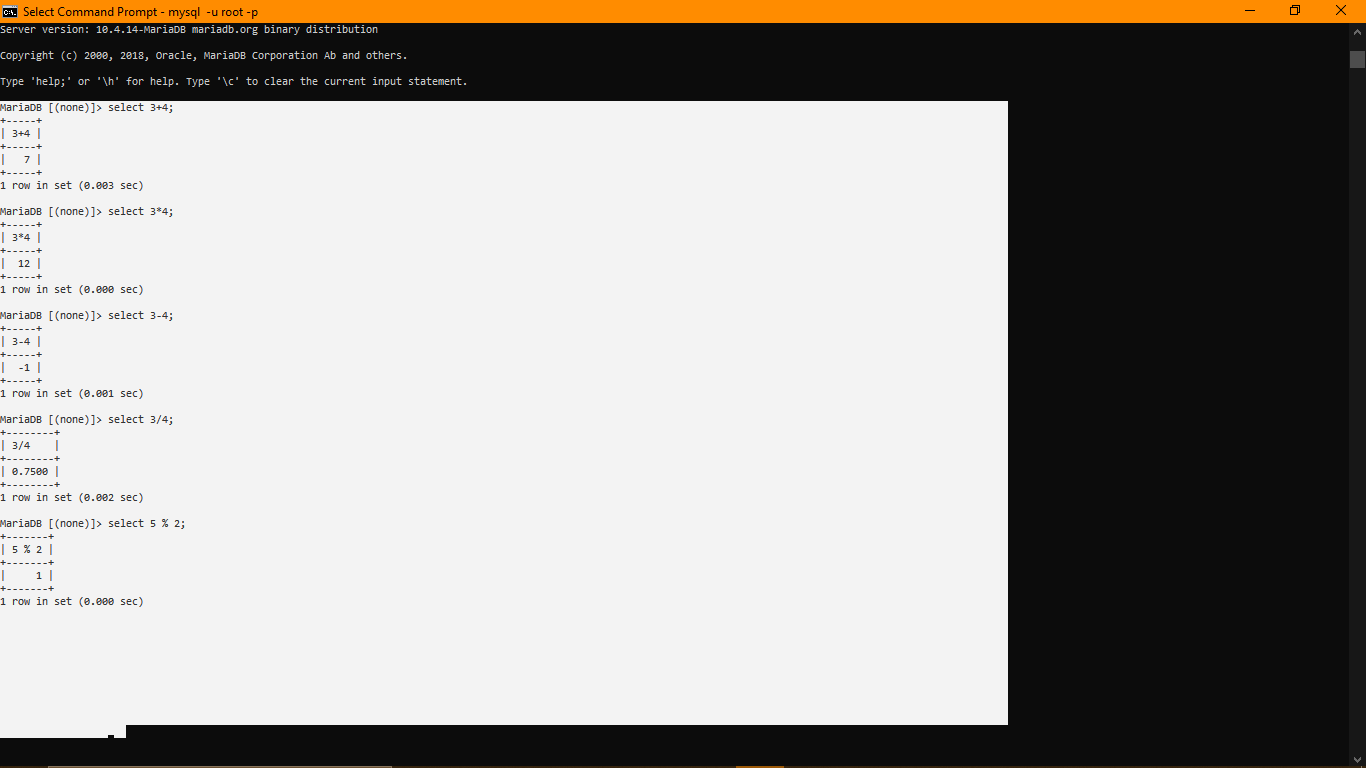
Hasil



MENAMPILKAN TANGGAL Gunakan perintah berikut untuk menampilkan tanggal. Pastikan akhir perintah di beri tanda TITIK KOMA (;) kemudian tekan enter.

MENAMPILKAN WAKTU ATAU JAM

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



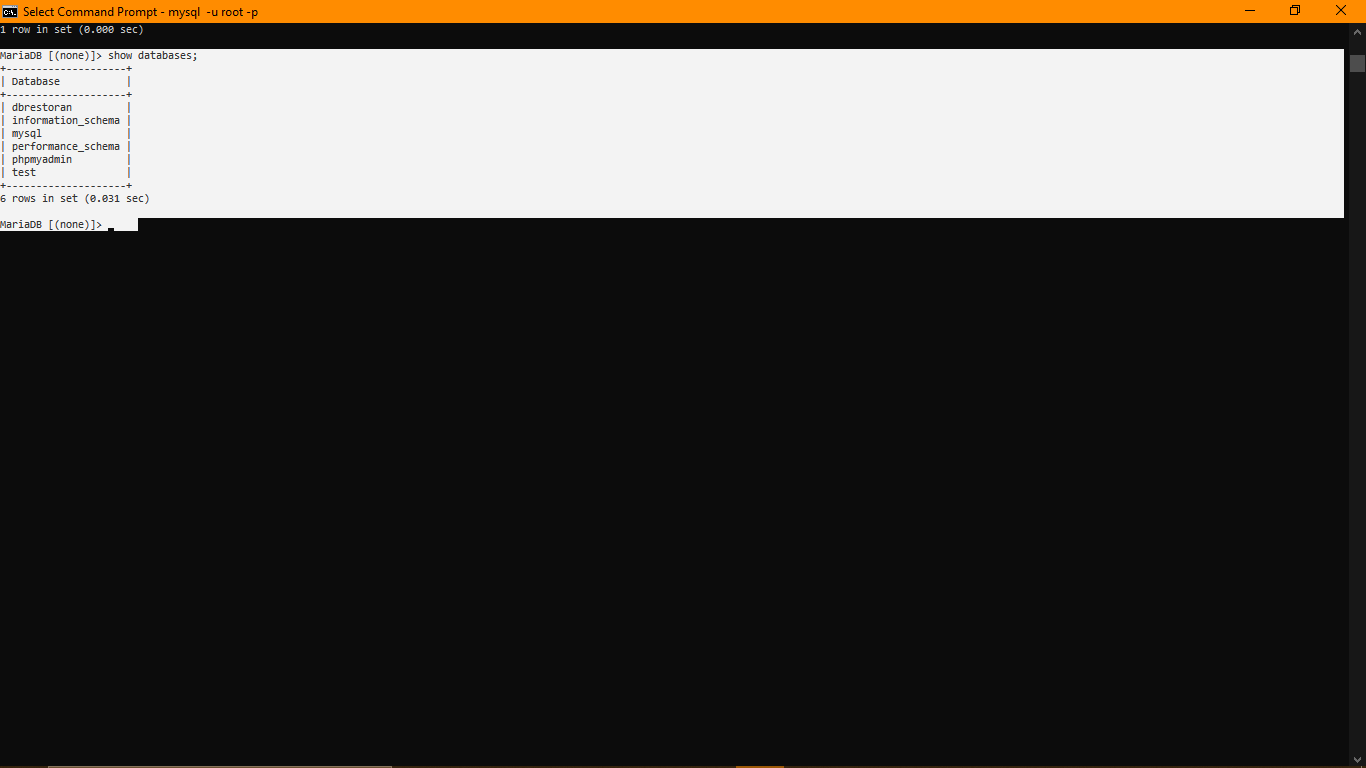
MENGGUNAKAN KALKULATOR Untuk menggunakan kalkulator di MySQL gunakan perintah sebagai diatas

**Data Definition Language Pembuatan Database, Tabel, & Index (ddl)**

DDL adalah proses pembuatan RUMAH dari data. Data akan disimpan pada KOLOM, kolom disimpan di TABEL. Tabel disimpan di DATABASE.

**MENAMPILKAN DATABASE**

**Untuk menampilkan semua database yang ada di MySQL komputer anda. Gunakan perintah; SHOW DATABASES;**

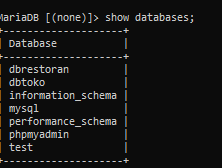


Hasil

**MEMBUAT DATABASE Pada latihan ini, akan membuat database dengan nama dbtoko. Gunakan perintah berikut; CREATE DATABASE nama\_database;**



Periksa database yang sudah dibuat dengan menggunakan perintah;



**MENGHAPUS DATABASE**

**Jika ada database yang tidak digunakan bisa dihapus dengan perintah; DROP DATABASE nama\_database;**



**MENGAKTIFKAN DATABASE**

**Jika database sebelumnya dihapus buat lagi dengan nama [dbtoko], gunakan perintah diatas. Sebelum membuat tabel anda harus memilih atau mengaktifkan database yang akan digunakan sebagai tempat penyimpanan tabel. Untuk memilih atau mengaktifkan gunakan perintah sebagai berikut;**

**USE nama\_database;**



Perhatikan terjadi perubahan dari [none] menjadi [dbtoko] ini menunjukan bahwa database yang aktif yang akan digunakan adalah [dbtoko];

TIPE DATA

sebelum membuat tabel anda harus mengetahui tipe data yang akan digunakan di dalam pembuatan tabel. TIPE DATA NUMERIK

Tipe data numerik berisi sekumpulan tipe data sejenis yang mampu menangani datadata numerik. Berikut ini beberapa tipe data yang digunakan dalam MySQL.

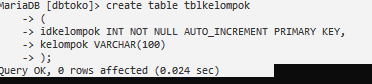
(nanti lanjutkan ji8ka perlu

**MEMBUAT TABEL**

Berikut adalah perintah untunk membuat table

**CREATE TABLE nama\_tabel (nama\_kolom TIPE\_DATA, nama\_kolom TIPE DATA);**

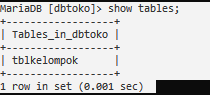
Pembuatan tabel berdasarkan desain yang sudah dibuat sebelumnya, tabel yang pertama dibuat adalah tabel tblkelompok



**MENAMPILKAN TABEL**

Untuk menampilkan tabel yang sudah dibuat, gunakan perintah berikut;

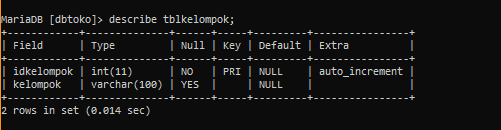
**SHOW TABLES;**



**MENAMPILKAN STRUKTUR TABEL**

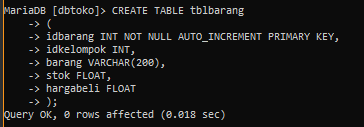
Untuk melihat struktur tabel gunakan perintah sebagai berikut;

**DESCRIBE nama\_tabel;**

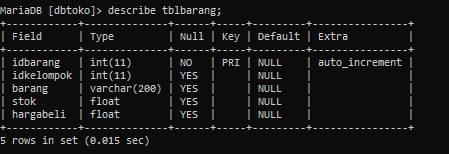


**Membuat Tabel tblbarang**

Untuk membuat tabel tblbarang gunakan perintah sebagai berikut;

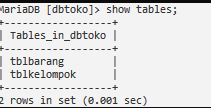


Periksa apakah struktur yang dibuat sudah benar



Menampilkan Tabel

Tabel yang sudah dibuat pada database bisa dilihat dengan menggunakan perintah berkut;



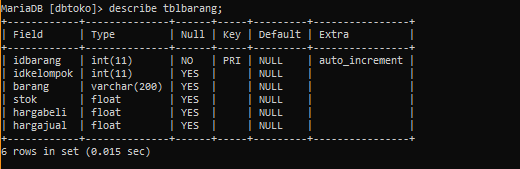
**MENAMBAH KOLOM**

Jika dilihat pada desain terdapat kekurangan kolom pada tabel yang dibuat, kita bisa menambahkan kolom dengan perintah sebagai berikut

**; ALTER TABLE nama\_tabel ADD nama\_kolom TIPE DATA;**



Setelah penambahan kolom periksa tblbarang dengan perintah sebagai berikut;



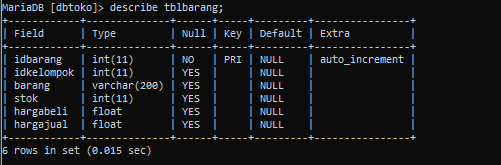
**MERUBAH TIPE DATA**

Pada tblbarang terdapat tipe data FLOAT pada kolom stok, kolom stok tipe data akan diganti menjadi INT. Gunakan perintah sebagai berikut;

**ALTER TABLE nama\_tabel MODIFY nama\_kolom TIPE DATA;**



Periksa kembali tipe data yang digunakan menggunakan perintah berikut;



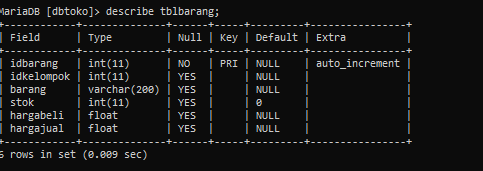
**MEMBERI NILAI DEFAULT PADA KOLOM**

Pada kolom [stok] tblbarang akan diberi nilai DEFAULT [0] atau nilai awal [0]. Gunakan perintah sebagai berikut; ALTER TABLE nama\_tabel ALTER COLUMN nama\_kolom

**SET DEFAULT ISI\_NILAI\_DEFAULT\_JIKA\_VARCHAR\_BERI\_TANDA\_PETIK;**

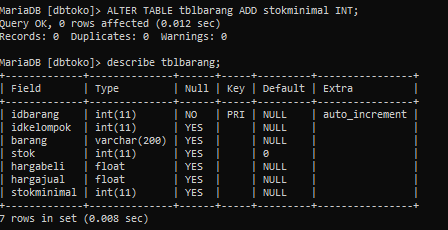


Periksa dengan menggunakan perintah berikut;



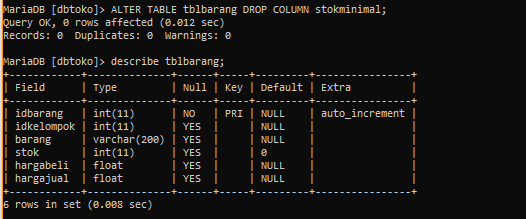
MENGHAPUS KOLOM

Pada latihan untuk menghapus kolom, tambahkan kolom baru dan check kolom tersebut dengan perintah sebagai berikut



Pada struktur diatas kolom stokminimal akan dihapus. Untuk menghapus gunakan perintah berikut; **ALTER TABLE nama\_tabel DROP COLUMN nama\_kolom;**

**Lalu perikasa table sebagai berikut**

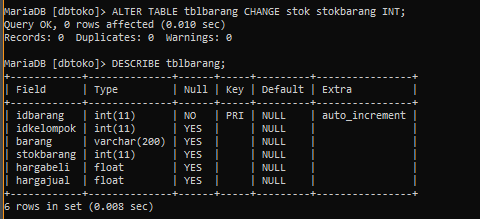


MERUBAH NAMA KOLOM

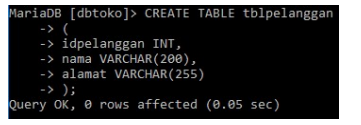
Untuk merubah nama kolom jika terdapat kesalahan bisa menggunakan perintah sebagai berikut;

**ALTER TABLE nama\_tabel CHANGE kolom\_lama kolom\_baru TIPE DATA;**

**Lalu perikasa struktur tabelnya**

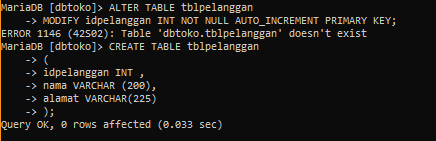


**MEMBUAT TABEL tblpelanggan**

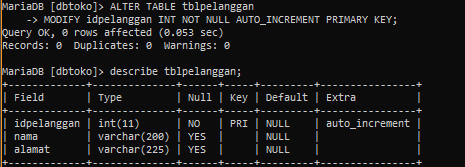


Pada pembuatan tabel pelanggan diatas belum ada PRIMARY KEY. Untuk menambahkan PRIMARY KEY gunakan perintah ubah tipe data.

**MENAMBAHKAN PRIMARY KEY**



Perbaikan pada pembuatan tabel diatas dengan merubah tipe data pada kolom dan menambahkan PRIMARY KEY; lalu priksa truktur table nya

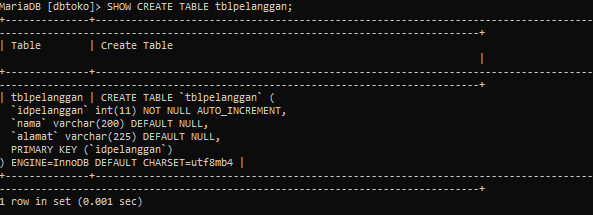


**MENAMPILKAN ENGINE YANG DIGUNAKAN**

Pada pembuatan database terdapat 2 engine atau mesin yang digunakan yaitu MYISAM dan InnoDB. Untuk proses pembuatan database relasional harus menggunakan InnoDB. MYISAM tidak bisa menerima penggunaan FOREIGN KEY.

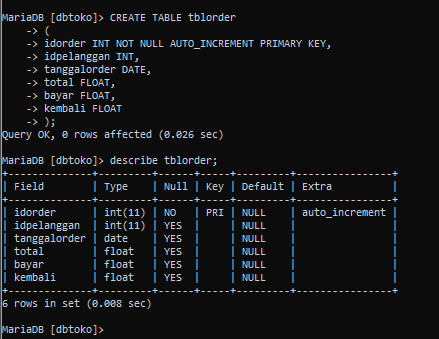
SHOW CREATE TABLE nama\_tabel;

Hasilnya



**MEMBUAT TABEL tblorder**

Perintah untuk membuat tblorder dan sturktur tabelnya

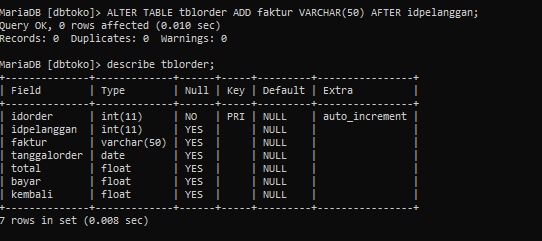


Perhatikan pada desain pembuatan tabel diatas terdapat kekurangan satu kolom yaitu kolom [faktur] yang letaknya dibawah kolom [idpelanggan].

**MENAMBAH KOLOM**

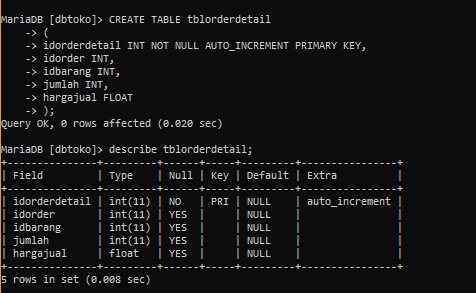
SETELAH KOLOM ALTER TABLE nama\_tabel ADD nama\_kolom TIPE DATA AFTER nama\_kolom\_sebelumnya;

Berikut hasilnya sekaligus struktur dari tablenya



MEMBUAT TABEL tblorderdetail

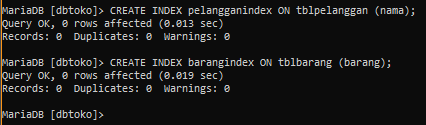
Perintah membuat tblorderdetail SEKALIGUS struktur tablenya



MEMBUAT INDEX

Index digunakan untuk mempercepat proses pencarian data terutama pada data yang jumlahnya diatas ribuan. Pada tabel diatas yang biasa digunakan untuk pencarian data adalah nama barang dan nama pelanggan.

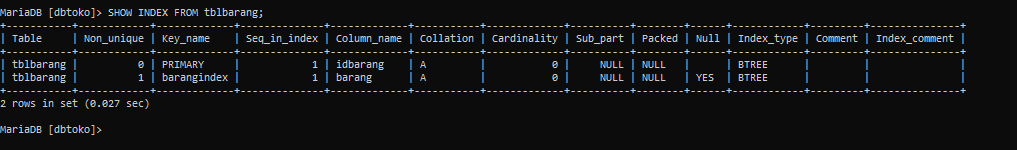
CREATE INDEX nama\_index ON nama\_tabel (kolom);



**MENAMPILKAN INDEX**

Index yang sudah dibuat bisa ditampilkan dengan perintah;

**SHOW INDEX FROM nama\_tabel;**



MENGHAPUS INDEX

Jika INDEX tidak digunakan bisa dihapus dengan perintah;

**DROP INDEX nama\_index ON nama\_tabel;**



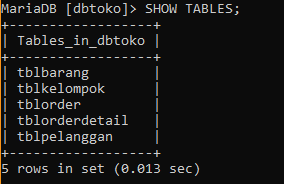
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Data Manipulation Language (DML)**

Insert, Delete, Update, Select, View, Trigger, Procedure, & Function

**INSERT SEMUA KOLOM**

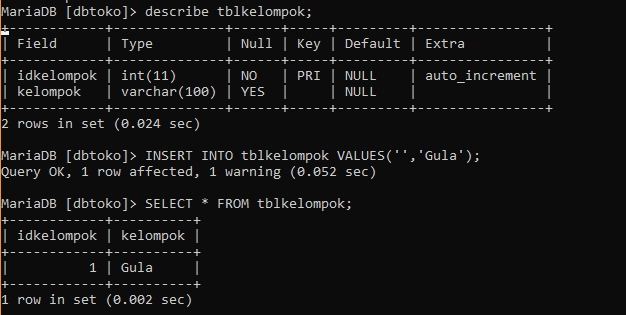
Insert semua kolom digunakan untuk memasukan data pada semua kolom tabel. Sebelum melakukan INSERT data periksa dulu tabel yang akan digunakan.



Sebagai latihan akan menggunakan tblkelompok, lihat dulu kolom tblkelompok

Pada tblkelompok terdapat 2 kolom yang akan di INSERT datanya, yaitu [idkelompok], [kelompok]. INSERT INTO nama\_tabel VALUES (kolom, kolom, semua\_kolom);

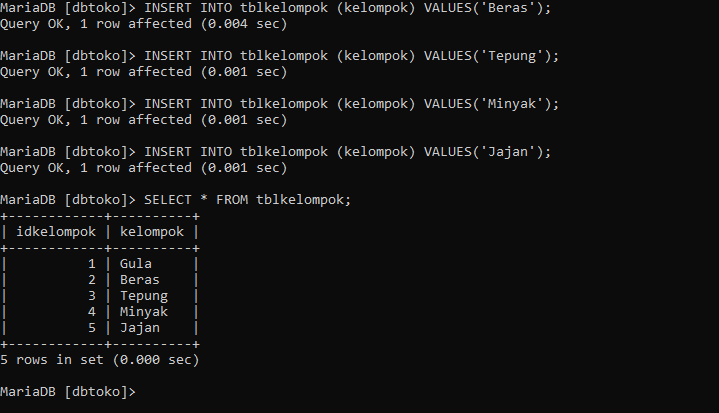
Lalu Periksa apakah data yang diinsert berhasil di gunakan



INSERT SEBAGIAN KOLOM INSERT

sebagian kolom digunakan untuk melakukan INSERT HANYA PADA kolom yang disebutkan.

INSERT INTO nama\_tabel (kolom\_yang\_disebut) VALUES (isi\_untuk\_semua\_kolom\_yang\_disebut);

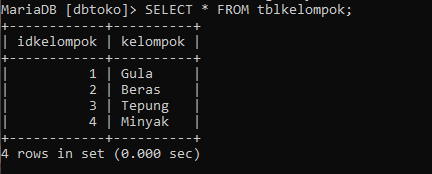


DELETE SEBAGIAN RECORD (BARIS DATA)

Untuk menghapus sebagian baris gunakan perintah;

DELETE FROM nama\_tabel WHERE baris\_yang\_akan\_dihapus;

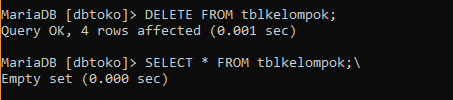




DELETE SEMUA RECORD

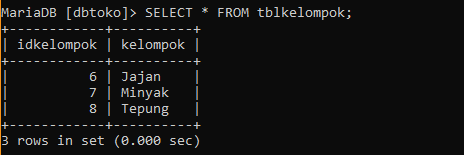
Untuk menghapus semua baris data bisa dilakukan dengan perintah;

DELETE FROM nama\_tabel;



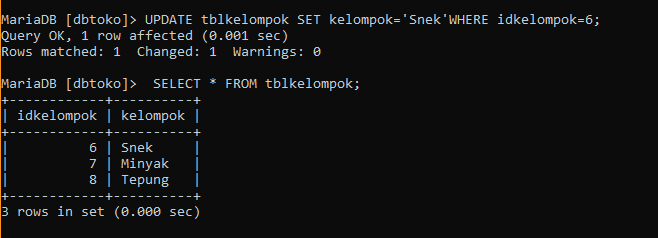
UPDATE SEBAGIAN RECORD

Karena semua data sudah dihapus semua pada materi delete, isi dulu data pada tblkelompok



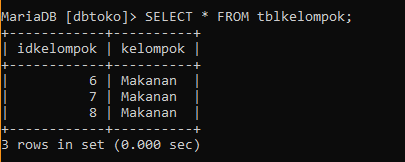
Pada tabel diatas UBAH data ‘Jajan’ menjadi ‘Snek’ menggunakan perintah sebagai berikut;

UPDATE nama\_tabel SET nama\_kolom=isi\_kolom WHERE nama\_kolom=baris\_yang\_dipilih\_untuk\_diubah;

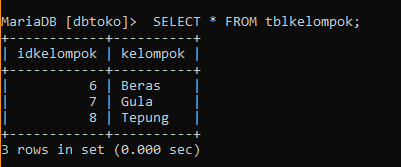


UPDATE SEMUA RECORD

Untuk mengubah semua data bisa dilakukan dengan perintah sebagai berikut; UPDATE nama\_tabel SET nama\_kolom=isi\_kolom;

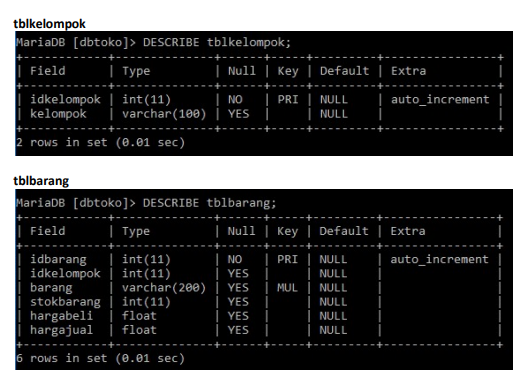


Lalu update lagi sebagai latihan



TABEL MASTER DAN TABEL TRANSAKSI (DETAIL)

Lihat struktur tblkelompok dan tblbarang;



**TABEL MASTER**

Tabel Master adalah tabel yang MEMASUKI tabel lain. Pada contoh diatas. Terdapat kolom [idkelompok] dari tabel [tblkelompok] yang MEMASUKI tabel [tblbarang]. Jadi tabel [tblkelompok] menjadi TABEL MASTER.

**TABEL TRANSAKSI**

Tabel Transaksi adalah tabel yang DIMASUKI tabel lain. Pada contoh diatas tabel [tblbarang] DIMASUKI kolom [idkelompok] yang berasal dari tabel [tblkelompok]. Jadi yang tabel [tblbarang] menjadi TABEL TRANSAKSI.

**PRIMARY KEY DAN FOREIGN KEY**

Pada tabel [tblbarang] terdapat kolom yang menjadi kunci atau KEY yang digunakan untuk membedakan baris data yang satu dengan dengan yang lain yaitu [iDCLrang]. Kolom yang berbeda antara baris data yang satu dengan yang lain disebut dengan PRIMARY KEY.

Pada tblbarang juga terdapat kolom yang masuk dari tabel lain yaitu [tblkelompok]. Kolom yang masuk ke tblbarang adalah [idkelompok]. Kolom yang masuk dari tabel lain disebut dengan FOREIGN KEY.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ELATIONAL DEPENDENCIES (HUBUNGAN KETERGANTUNGAN)**

DATABASE dibuat agar pengaturan data benar sesuai dengan yang diharapkan. Relational Dependencies adalah pengaturan agar tabel yang satu terhubung dengan tabel yang lain. Sehingga data yang masuk ke satu tabel bergantung dari tabel yang lain dalam bentuk BER-URUTAN

Yang pertama kali bisa di isi adalah tabel [tblkelompok] dan [tblpelanggan]. Setelah tabel [tblkelompok] di isi maka tabel [tblbarang] baru bisa di isi. Jadi tabel [tblbarang] TERGANTUNG (DEPENDENCIES) dari tabel [tblkelompok]. Dibuat demikian agar setiap barang yang masuk pada tabel [tblbarang] mempunyai kelompok.

[tblorder] baru bisa di isi jika tabel [tblpelanggan] SUDAH di isi. Maka tabel [tblorder] TERGANTUNG pada tabel [tblpelanggan] sehingga hanya pelanggan yang sudah masuk pada tabel [tblpelanggan] saja yang bisa melakukan order.

[tblorderdetail] baru bisa di isi jika [tblorder] dan tabel [tbbarang] SUDAH di isi. Maka tabel [tblorderdetail] TERGANTUNG pada tabel [tblorder] dan [tblpelanggan]. Jadi hanya pelanggan yang sudah melakukan order saja yang bisa diambilkan barangnya. Ini semua disebut dengan RELATIONAL DEPENDENCIES (HUBUNGAN KETERGANTUNGAN).

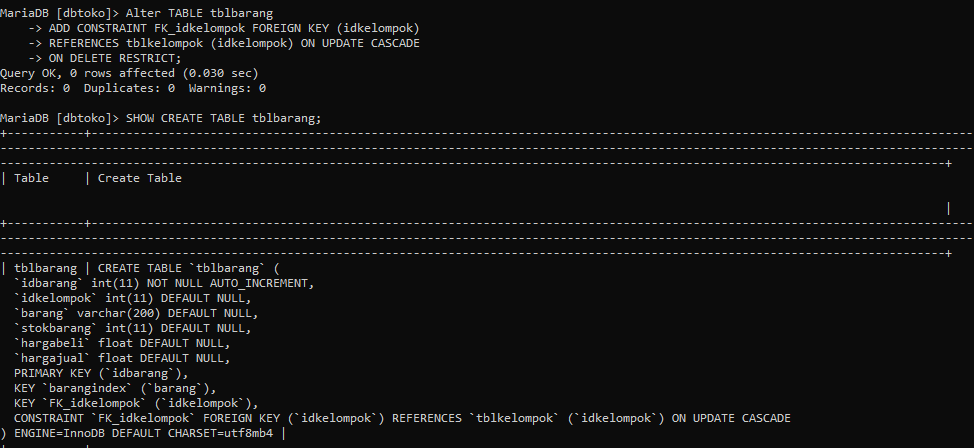
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**MEMBUAT RELASI ANTAR TABEL**

Lihat dulu struktur tabel yang akan dibat relasinya

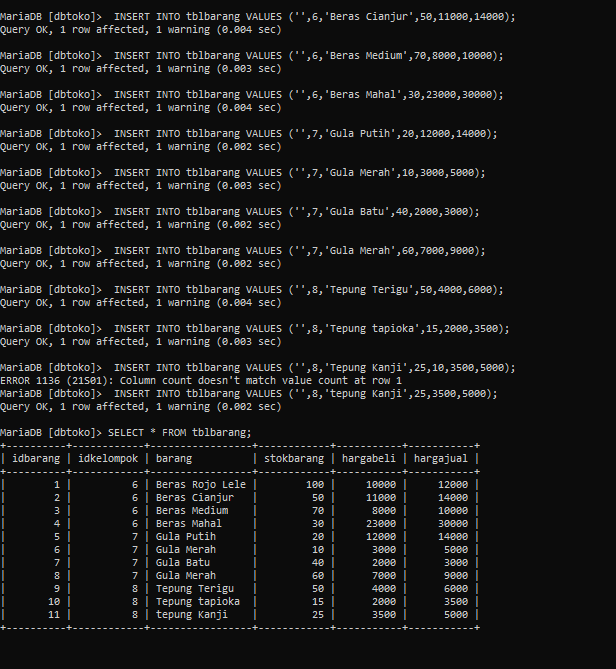
Perintah untuk membuat relasi adalah sebagai berikut

; ALTER TABLE nama\_tabel\_detail ADD CONSTRAINT FK\_foreign\_key\_pada\_tabel\_detail FOREIGN KEY (foreign\_key) REFRENCES nama\_tabel\_master (primary\_key\_tabel\_master) ON UPDATE CASCADE ON DELETE RESTRICT ;



**MENGUJI HASIL PEMBUATAN RELASI**

Untuk menguji hasil pembuatan relasi lakukan INSERT data pada tabel master terlebih dahulu baru kemudian ke tabel transaks



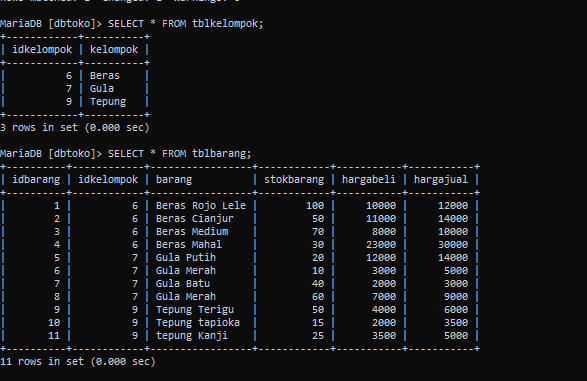
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PENGUJIAN INSERT

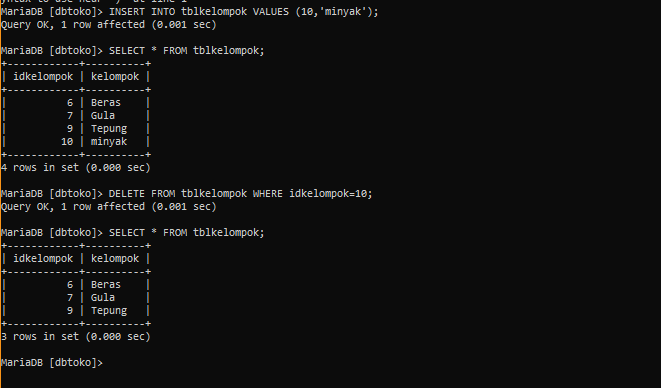
Insert pada tabel [tblbarang] HANYA BISA dilakukan menggunakan data yang tersedia pada tabel master. [Idkelompok] pada tabel master yang tersedia hanya [6,7,8].. Perhatikan hasil pengujian berikut; Pada pengujian INSERT idkelompok yang digunakan adalah 9 yang TIDAK TERSEDIA pada tabel master [tblkelompok].

PENGUJIAN UPDATE

Coba lakukan update pada tabel master kemudian periksa di tabel transaksi.



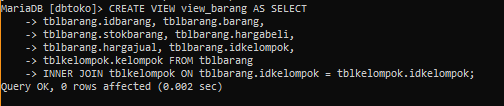
PENGUJIAN DELETE

Pengujian delete dilakkan dengan menghapus tabel master

PEMBUATAN VIEW

View digunakan untuk menggabungkan tabel master dan tabel transaksi menjadi sebuah tabel baru agar lebih mudah dilihat dan digunakan.

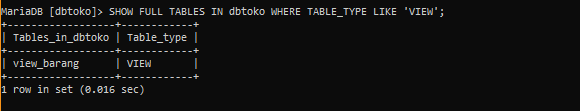
CREATE VIEW nama\_view AS SELECT tabel\_transaksi.kolom\_transaksi, tabel\_master.kolom\_master FROM tabel\_transaksi INNER JOIN tabel\_master ON tabel\_transaksi.kolom\_transaksi = tabel\_master.kolom\_master;



MENAMPILKAN SEMUA VIEW YANG SUDAH DIBUAT

View yang sudah dibuat bisa dilihat dengan perintah;

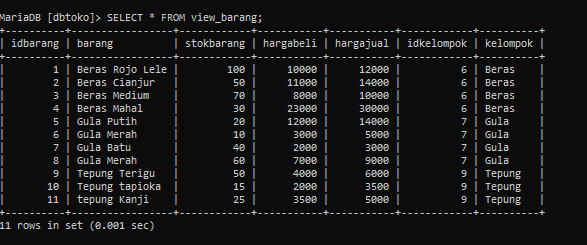
SHOW FULL TABLES IN nama\_database WHERE TABLE\_TYPE LIKE ‘VIEW’;



MELIHAT ISI VIEW

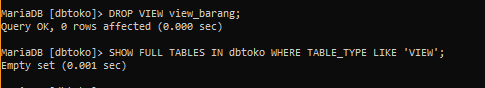
Untuk melihat isi VIEW sama dengan melihat isi tabel

SELECT \* FROM nama\_view;



MENGHAPUS VIEW

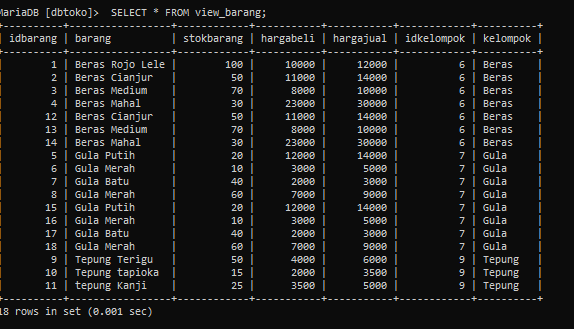
Jika VIEW yang sudah dibuat tidak digunakan lagi bisa di hapus dengan menggunakan perintah berikut; DROP VIEW nama\_view;



SELECT SEMUA KOLOM (\*)

Select semua adalah menampilkan semua kolom dari tabel atau VIEW

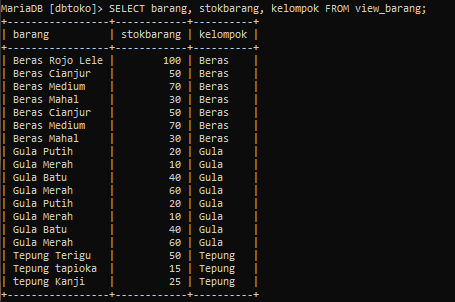
SELECT \* FROM nama\_tabel\_atau\_nama\_view;



SELECT SEBAGIAN KOLOM

Select sebagian kolom adalah menampilkan hanya pada kolom yang dipilih

SELECT nama\_kolom, nama\_kolom FROM nama\_tabel\_atau\_nama\_view;

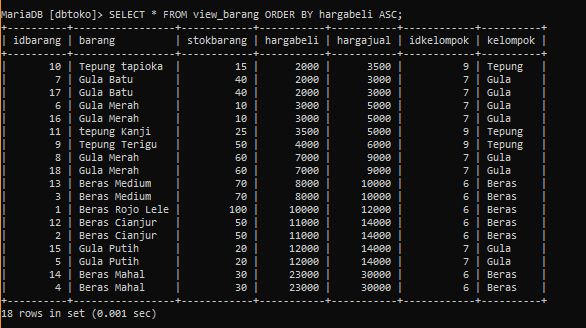


**SELECT ORDER**

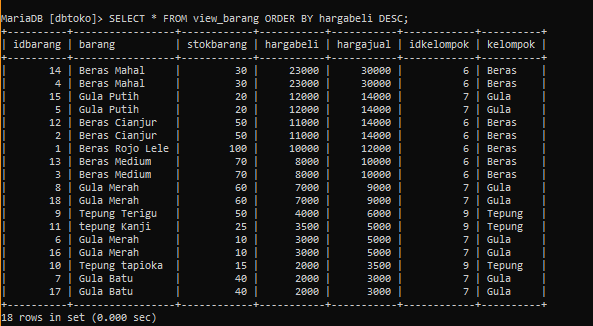
SELECT ORDER adalah cara untuk menampilkan data dalam urutan naik atau turun Jika NAIK menggunakan ASC Jika TURUN menggunakan

DESC SELECT \* FROM nama\_tabel\_atau\_view ORDER BY nama\_kolom ASC;

Hasil Select Order ASC



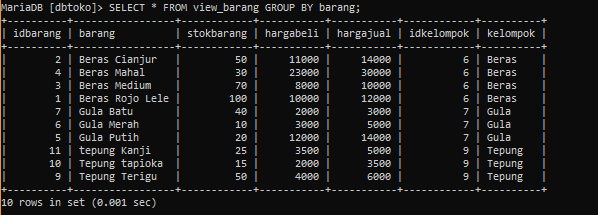
Hasil Select Order DESC



SELECT GROUP

Perintah SELECT GROUP digunakan untuk mengelompokan data sesuai dengan kelompok dari kolom yang dipilih.

SELECT \* FROM nama\_tabel\_atau\_view GROUP BY nama\_kolom;



PENGUJIAN WHERE

Pengujian WHERE pengujian pada SQL yang bisa diletakan di belakang:

• SELECT

• DELETE

• UPDATE

Pengujian akan menjalankan SELECT, DELETE, UPDATE jika kondisi yang diuji terpenuhi

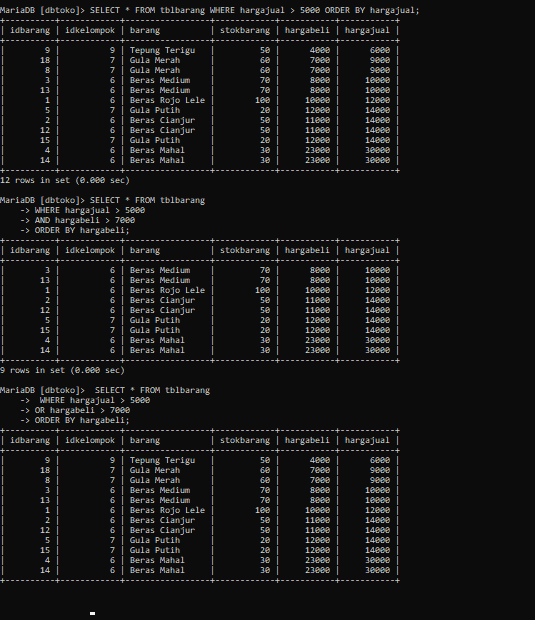
• Operator pembanding [=, >, <>, >=, <=, LIKE]

• Operator LOGIKA (AND, OR)

• SELECT

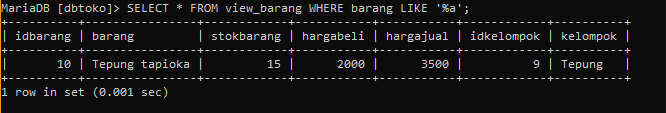
Contoh pengujian

SELECT \* FROM nama\_tabel\_atau\_view WHERE pengujian ORDER BY nama\_kolom;

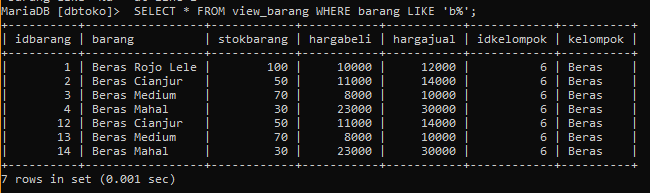


SELECT \* FROM nama\_tabel\_atau\_view WHERE nama\_kolom LIKE ‘%\_apapun’;

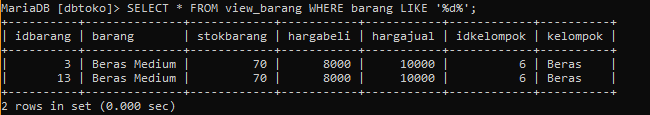
% adalah sebutan untuk sembarang atau apapun Contoh: %a artinya awalan sembarang yang penting akhiranya a



b% artinya awalan b akhiran sembarang



%d% artinya awalan sembarang dan akhiran sembarang yang tengahnya ada huruf d



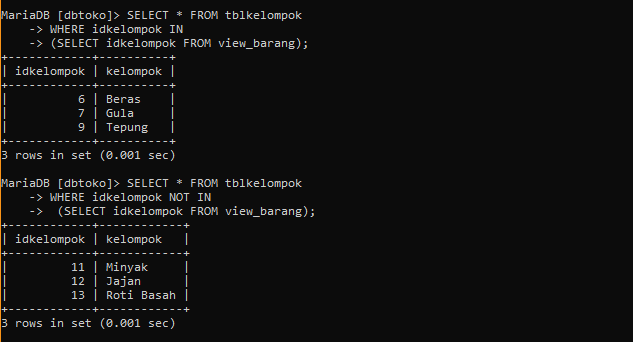
**SUBQUERY (SELECT IN SELECT)**

Subquery adalah SELECT yang ada di dalam SELECT

Pada [view\_barang] ada kelompok yang sudah menggunakan data dari tabel [tblkelompok]. Ada pertanyaan yang muncul dari data diatas adalah:

1. Ada berapa data dari tabel [tblkelompok] yang SUDAH digunakan pada VIEW?

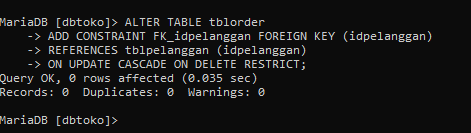
2. Ada berapa data yang BELUM digunakan pada VIEW? Untuk menjawab pertanyaan diatas bisa menggunakan SUBQUERY. SELECT \* FROM nama\_tabel WHERE nama\_kolom IN (SELECT nama\_kolom FROM nama\_tabel)



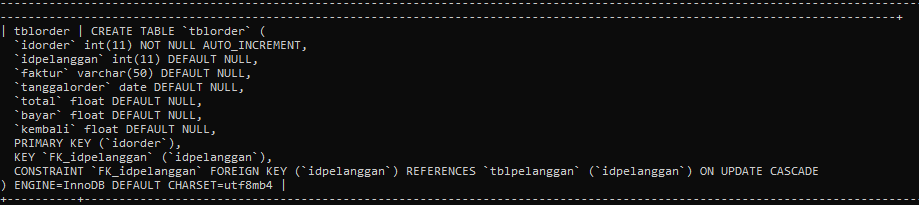
**MEMBUAT RELASI ANTAR TABEL**

PROSES BISNIS

Pelanggan yang akan membeli akan dicatat di tabel order. Hanya pelanggan yang sudah terdaftar pada tabel pelanggan yang bisa melakukan order.

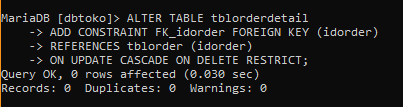


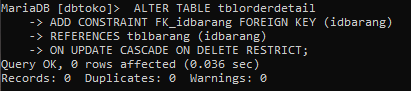
Periksa hasil pembuatan RELASI



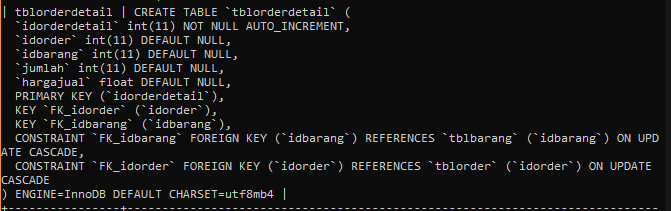
**RELASI 2 TABEL MASTER DAN 1 TABEL TRANSAKSI**

Buat relasi antara tabel [tblorderdetail] dan tabel [tblorder] dan juga t relasi antara tabel [tblorderdetail] dan tabel [tblbarang]



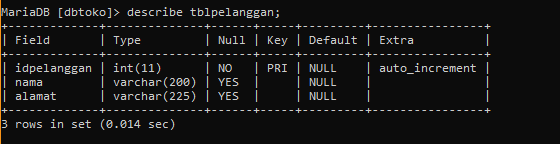


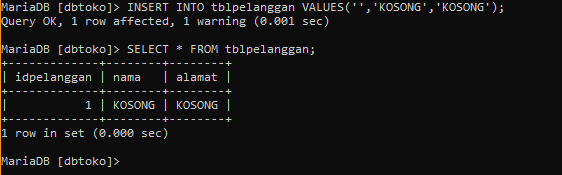
Hasil



DUMMY DATA

Dummy data adalah data yang tidak sebenarnya. Data ini diperlukan jika data sebenarnya tidak bisa diperoleh. Dummy data biasa digunakan selama pengujian database.





Pada tabel terdapat nama pelanggan dengan nama [KOSONG], jika terjadi pembelian dengan nama pelanggan yang tidak disebutkan maka aplikasi akan memberi nama pelanggan tersebut dengan nama [KOSONG]. Nama pelanggan dengan nama [KOSONG] ini disebut dengan dummy data.

KONSEP TRIGGER

Trigger adalah perintah INSERT, UPDATE, DELETE, FUNCTION, PROCEDURE yang ditanam pada MySQL yang akan DIJALANKAN pada kejadian berikut:

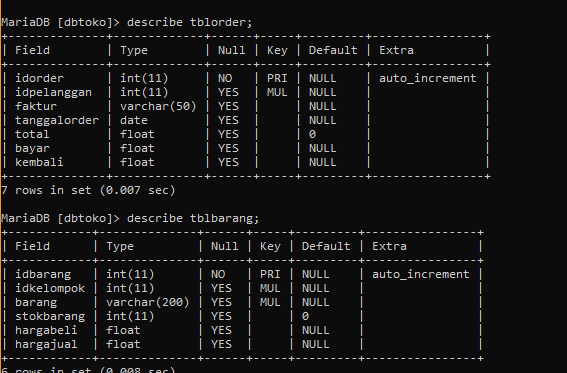
• AFTER INSERT (setelah INSERT) pada tabel yang dimaksud

• BEFORE INSERT (sebelum INSERT) pada tabel yang dimaksud

• AFTER DELETE (setelah DELETE) pada tabel yang dimaksud • BEFORE DELETE (Sebelum DELETE) pada tabel yang dimaksud

• AFTER UPDATE (Setelah UPDATE) pada tabel yang dimaksud

• BEFORE UPDATE (Sebelum UPDATE) pada tabel yang dimaksud Sebelum membuat trigger pastikan anda memahami proses bisnis dari pembuatan database yang anda lakukan



**KONSEP TRIGGER**

Trigger adalah perintah INSERT, UPDATE, DELETE, FUNCTION, PROCEDURE yang ditanam pada MySQL yang akan DIJALANKAN pada kejadian berikut:

• AFTER INSERT (setelah INSERT) pada tabel yang dimaksud

• BEFORE INSERT (sebelum INSERT) pada tabel yang dimaksud

• AFTER DELETE (setelah DELETE) pada tabel yang dimaksud

• BEFORE DELETE (Sebelum DELETE) pada tabel yang dimaksud

• AFTER UPDATE (Setelah UPDATE) pada tabel yang dimaksud

• BEFORE UPDATE (Sebelum UPDATE) pada tabel yang dimaksud

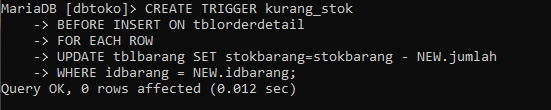
**PEMBUATAN TRIGGER**

|  |  |  |
| --- | --- | --- |
| Nama Trigger | Kejadian | Fungsi |
| kurang\_stok | BEFORE INSERT ON | Mengurangi [stokbarang] pada tabel [tblbarang] SEBELUM INSERT pada tabel [tblorderdetail] |
| tambah\_total | AFTER INSERT ON | Menambah [total] pada tabel [tblorder] SESUDAH INSERT pada tabel [tblorderdetail] |
| tambah\_stok | BEFORE DELETE ON | Manambah [stokbarang] pada tabel [tblbarang] SEBELUM DELETE pada tabel [tblorderdetail] |
| kurang\_total | AFTER DELETE ON | Mengurangi [total] pada tabel [tblorder] SESUDAH DELETE pada tabel [tblorderdetail] |

**CODING:**

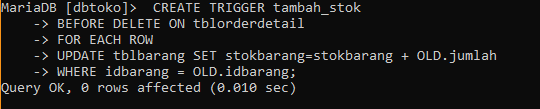
CREATE TRIGGER nama\_trigger AFTER INSERT ON nama\_tabel\_yang\_dipasang\_trigger FOR EACH ROW ketik\_perintah\_trigger\_disini\_akhiri\_dengan\_titik\_koma;

**kurang\_stok**



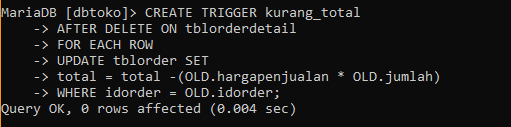
**PENJELASAN: Trigger [kurang\_stok] digunakan untuk mengurangi [stokbarang] paad [tblbarang] pada saat BEFORE INSERT (sebelum insert) pada tabel [tblorderdetail]. NEW.jumlah adalah data atau nilai yang AKAN MASUK pada kolom [jumlah] di table** [tblorderdetail]

**tambah\_total**



PENJELASAN Trigger [tambah\_stok] digunakan untuk menambah [stokbarang] pada [tblbarang] pada saat BEFORE DELETE (sebelum delete) pada tabel [tblorderdetail]. OLD.jumlah adalah data yang SUDAH ADA pada kolom [jumlah] di tabel [tblorderdetail]

**kurang\_total**



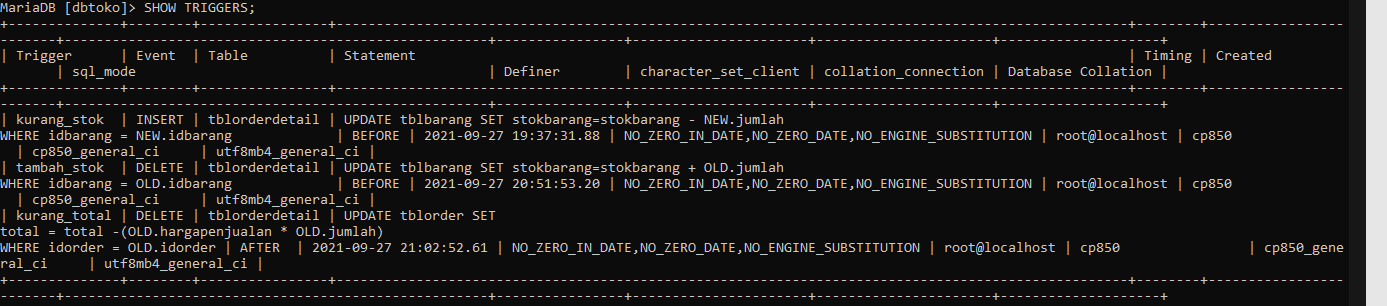
PENJELASAN: Trigger [kurang\_total] digunakan untuk mengurangi nilai [total] pada tabel [tblorder] saat AFTER DELETE (sesudah DELETE) pada tabel [tblorderdetail].

MENAMPILKAN TRIGGER

Trigger yang sudah dibuat bisa ditempilkan dengan cara



**Hasil:**



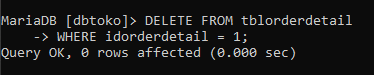
**PENGUJIAN TRIGGER**

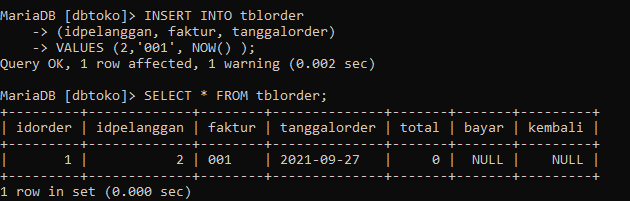
Setelah trigger dibuat maka langkah selanjutnya adalah menguji trigger;

**PENGUJIAN INSERT PADA TABEL**

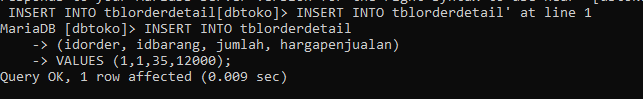
[tblorderdetail] Buat order atas nama pelanggan [smkrevit] dengan [idpelanggan = 2] pada tabel [tblorder] dengan cara memasukan data sebagai berikut;

Buat order atas nama pelanggan [smkrevit] dengan [idpelanggan = 2] pada tabel [tblorder] lalu periksa

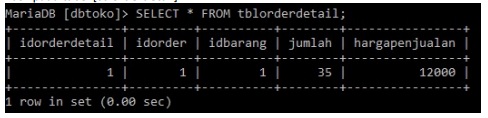
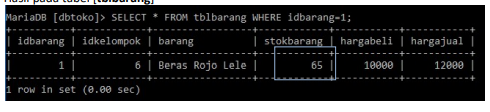


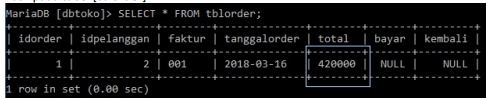


buat INSERT data pada tabel [tblorderdetail] dengan menggunakan [idorder = 1] sesuai yang ada di tabel [tblorder] dengan [iDCLrang = 1] pada tabel [tblbarang].

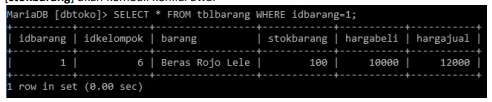


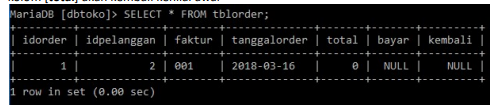
HASIL:





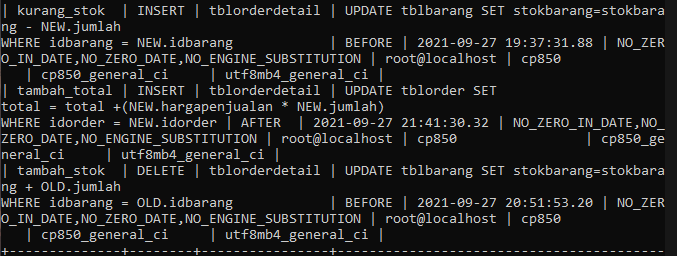
PENGUJIAN DELETE PADA TABEL [tblorderdetail]



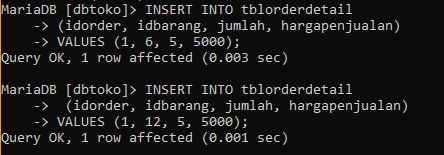


HAPUS TRIGGER

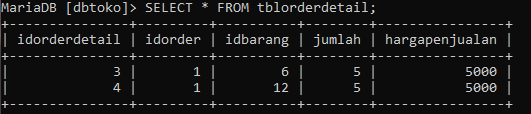




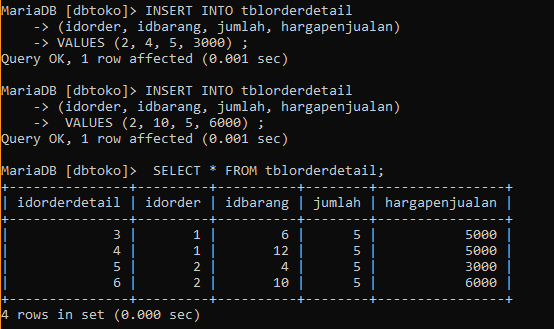
JOIN (GABUNGAN TABEL)



PERIKSA tabel [tblorderdetail]

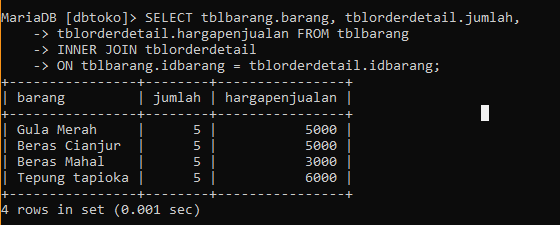


Lakukan insert data pada [tblorderdetail] Lalu periksa hasilnya

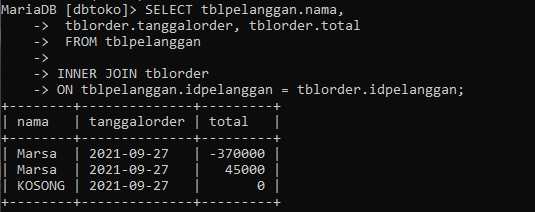


**INNER JOIN (MENGAMBIL BAGIAN YANG ADA DI TABEL MASTER DAN TABEL TRANSAKSI)**

JOIN HARUS DIMULAI DARI TABEL MASTER

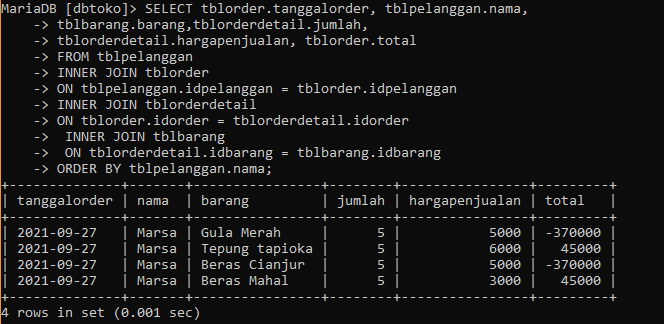


**INNER JOIN untuk mengetahui pelanggan yang melakukan ORDER**



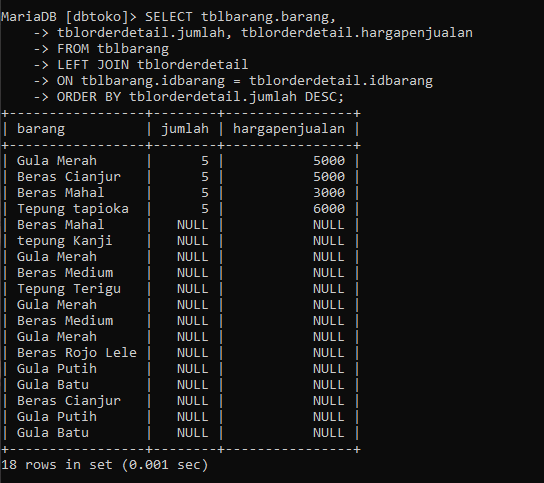
INNER JOIN BANYAK TABEL

Dengan menggunakan INNER JOIN banyak tabel bisa diketahui barang yang dibeli pelanggan dan total pembelian oleh pelanggan.



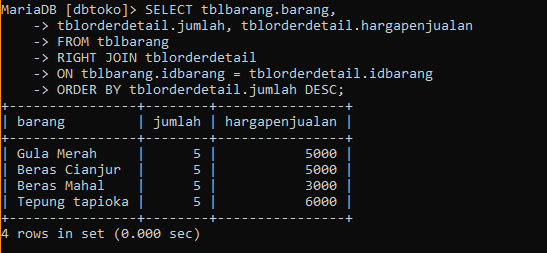
LEFT JOIN (MENAMPILKAN TABEL MASTER)

Left join digunakan untuk menampilkan semua yang ada di tabel master dan mengambil sebagian yang ada di tabel transaksi



RIGHT JOIN (MENAMPILKAN TABEL TRANSAKSI)

Right Join digunakan untuk menampilkan semua yang ada di tabel transaksi dan sebagian yang ada di tabel master.

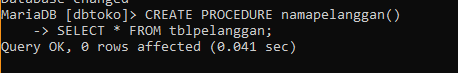


STORE PROCEDURE

Stor Procedure adalah blok program yang diletakan pada MySQL. Blok yang sudah dibuat bisa dipanggil jika diperlukan. PROCEDURE TIDAK MEMILIKI RETURN

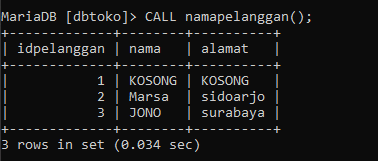
PROCEDURE TANPA PARAMETER

CREATE PROCEDURE nama\_procedure Isi\_procedure;



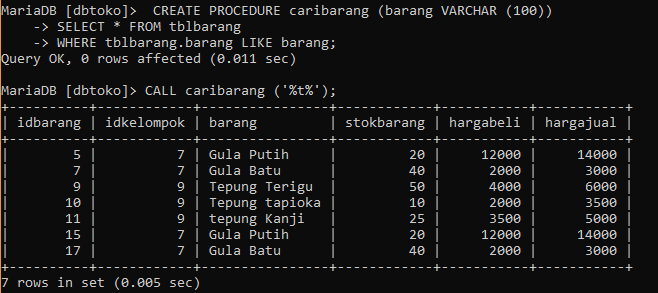
PEMANGGILAN PROCEDURE

CALL nama\_procedure;



PROCEDURE DENGAN PARAMETER

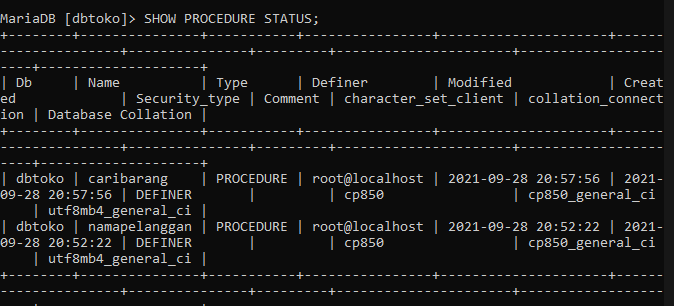
Panggil procedure yang sudah dibuat dengan parameter yang dimasuka [‘%t%’] yang menunjukan nama barang yang mengandung huruf [t]



Dengan menggunakan PROCEDURE kita bisa membuat semua QUEREY atau SELECT di dalam procedure sehingga memudahkan dalam pembuatan aplikasi.

MENAMPILKAN SEMUA PROCEDURE

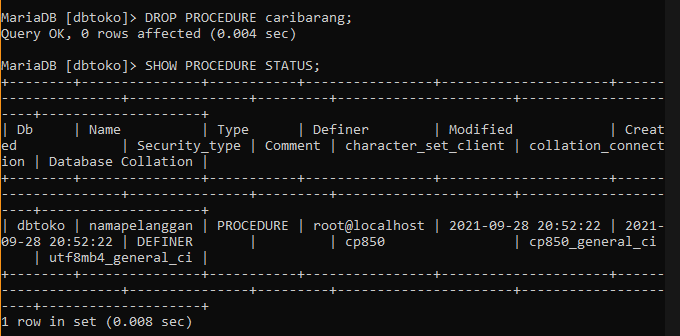
MySQL bisa menampilkan semua isi procedure dengan perintah berikut;



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

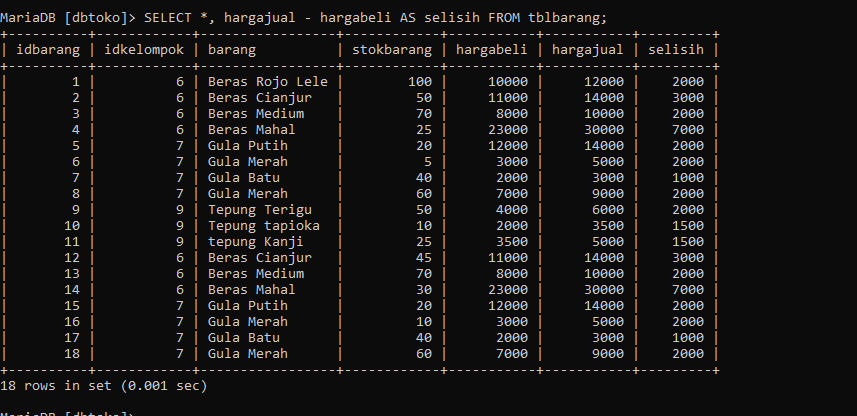
MENGHAPUS STORE PROCEDURE

Untuk menghapus store procedure bisa menggunakan perintah;

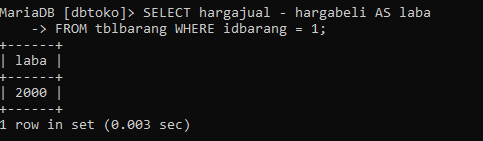


FUNCTION

Function adalah blok program yang disimpan di MySQL yang bisa menerima INPUT atau PAREMETER dan MEMILIKI RETURN

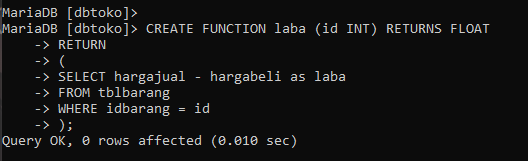


Selisih [hargajual] dan [hargabeli] ini disebut dengan [laba]. Sekarang akan diambil selisih perbarang untuk dihitung sebagai laba. Sebagai contoh diambil satu barang dengan [iDCLrang = 1] maka selisihnya akan ditampilkan seperti gambar dibawah.



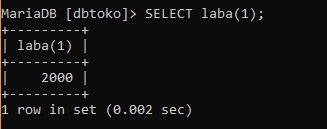
CREATE FUNCTION nama\_function (parameter TIPE DATA) RETURNS TIPE DATA RETURN (isi\_ function);

Isi parameter tipe data sesuai dengan data yang dimasukan. Pada contoh diatas parameter yang dimasukan adalah [iDCLrang] dengan tipe data INT. Output yang dihasilkan dari isi function diatas adalah selisih [hargajual] – [hargabeli] yang bertipe data FLOAT;



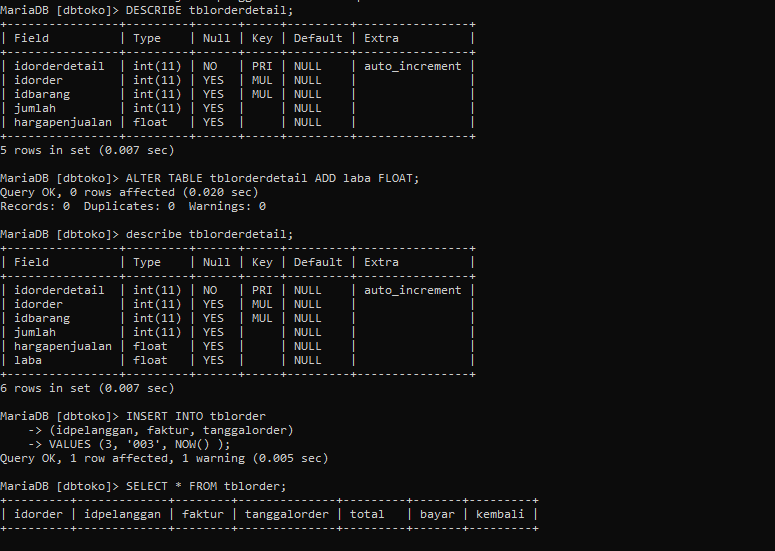
MEMANGGIL FUNCTION

Untuk memanggil function gunakan perintah berikut; SELECT FUNCTION nama\_function (parameter);



PENGGUNAAN FUNCTION

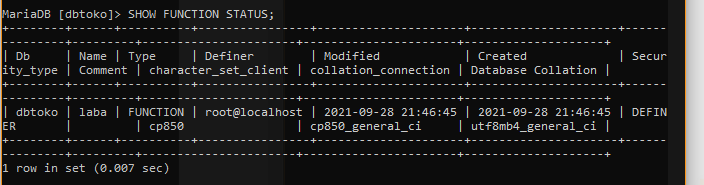
Materi berikut akan menunjukan penggunaan function pada aplikasi yang sedang dibuat. Lihat struktur tabel [tblorderdetail]



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

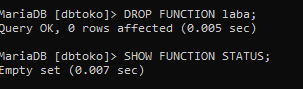
MENAMPILKAN SEMUA FUNCTION

Untuk menampilkan function yang sudah dibuat bisa menggunakan



MENGHAPUS FUNCTION

Untuk menghapus function bisa menggunakan perintah berikut;

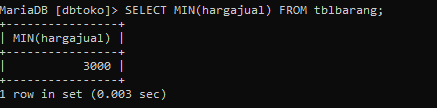


SELECT AGGREGATE

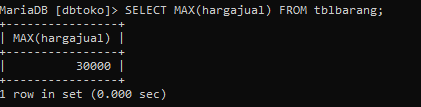
SELECT AGREGATE adalah SELECT yang menampilkan function bawaan dari MySQL. Ada beberapa select aggregate yang akan dipelajari yaitu;

**SELECT function\_aggregate(nama\_kolom); ANTARA NAMA AGGREGATE DAN KURUNG TIDAK BOLEH ADA SPASI**

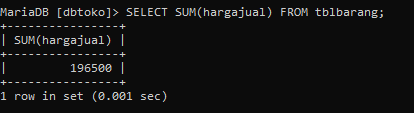
SELECT MIN



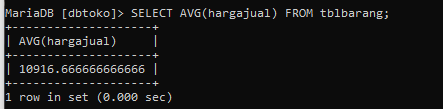
SELECT MAX



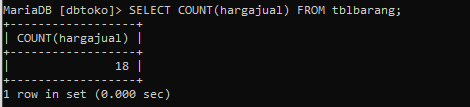
**SELECT SUM**



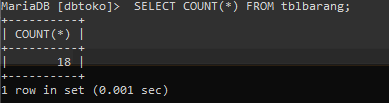
**SELECT AVG**



**SELECT COUNT**



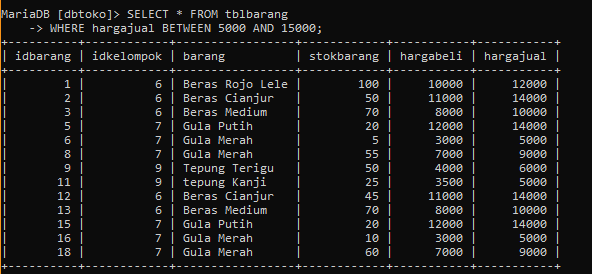
**SELECT COUNT(\*)**



SELECT BETWEEN (SELECT ANTARA DUA NILAI)

Select yang digunakan untuk menampilkan data antara dua nilai terendah dan tertinggi;

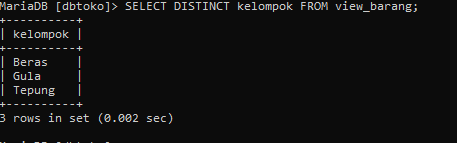
SELECT nama\_kolom\_atau\_\* FROM nama\_tabel WHERE nama\_kolom BETWEEN awal AND akhir



SELECT DISTINCT (MENAMPILKAN DATA YANG SAMA HANYA SATU KALI)

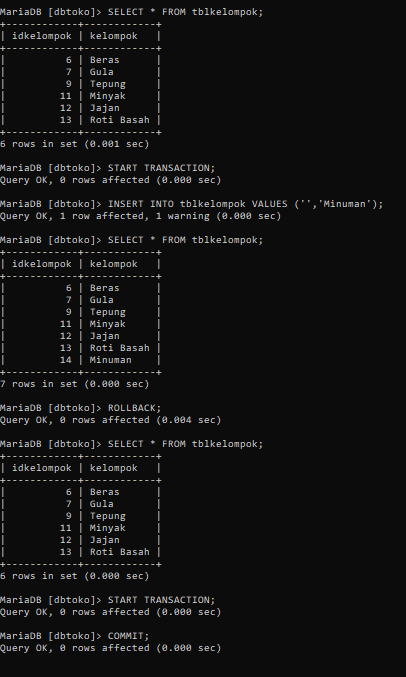
Untuk menampilkan data yang sama satu kali.

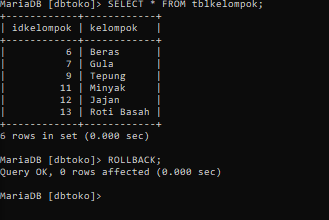
SELECT DISTINCT nama\_kolom FROM nama\_tabel\_view



START TRANSACTION, COMMIT, DAN ROLLBACK

Setiap perintah di MySQL yang masuk kelompok (INSERT, UPDATE, DELETE, SELECT) bisa dimasukan kedalam perintah START TRANSACTION, COMMIT, DAN ROLLBACK. Perintah yang dimulai dengan START TRANSACTION bisa dilakukan ROLLBACK (undo atau pembatalan perintah). COMMIT digunakan agar perintah yang dijalankan TIDAK BISA DI ROLLBACK (undo). buka kembali tabel [tblkelompok];





Untuk menambah user gunakan perintah berikut;

CREATE USER ‘nama\_user’@’host’;

