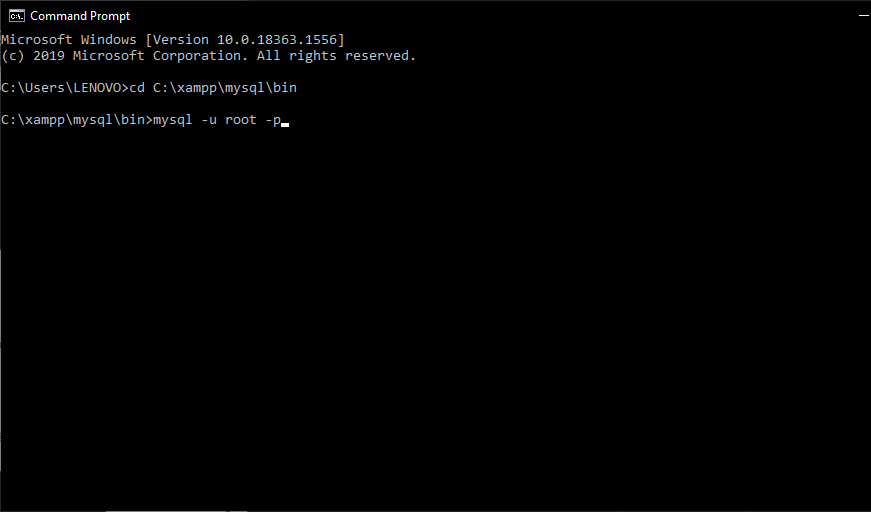
**FORM PENILAIAN MANDIRI**

**PEMRROGRAMAN WEB DAN MOBILE**

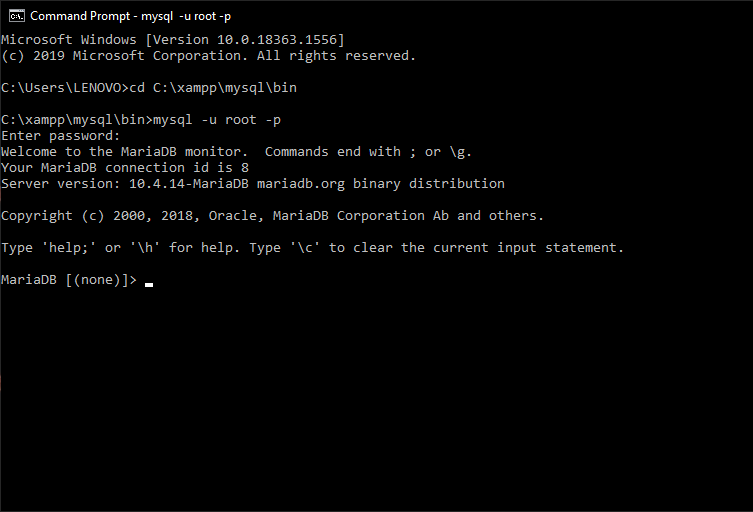
|  |  |
| --- | --- |
| **Absen** | **Nama** |
| 11 | Gazzak Cakrawartya |

|  |  |
| --- | --- |
| Materi | Nilai |
| BASIS DATA Buku PDF Mempelajari MySQL dari 0 Minggu 11 | 83 |

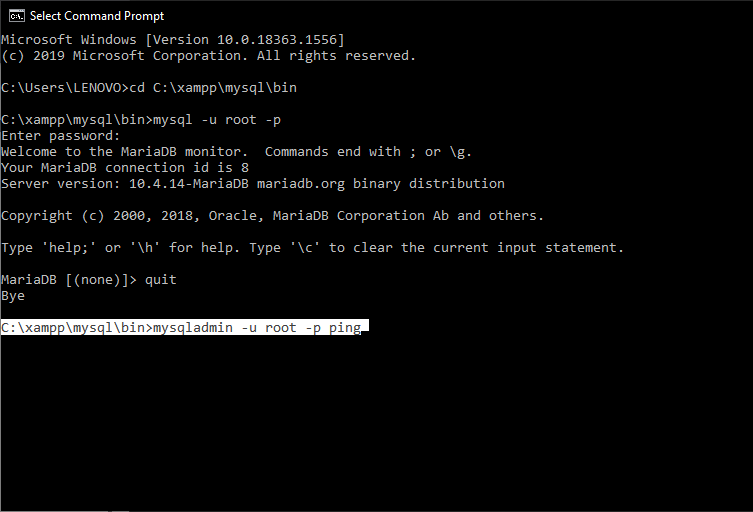
**DDL**



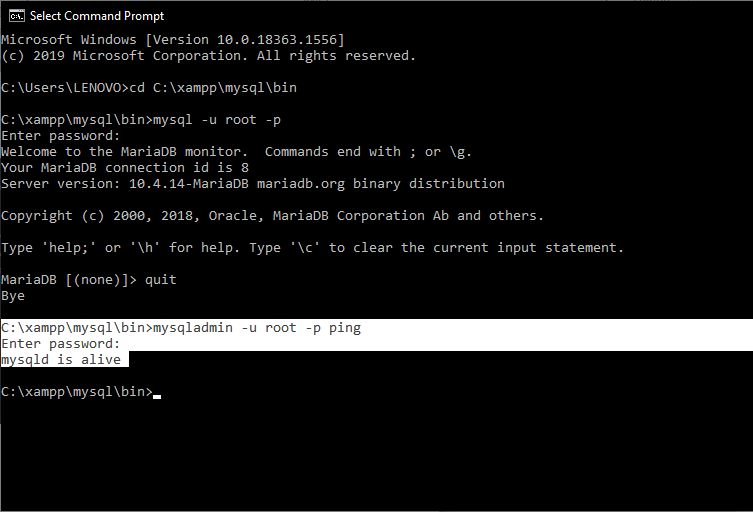
Menjalankan MySQL



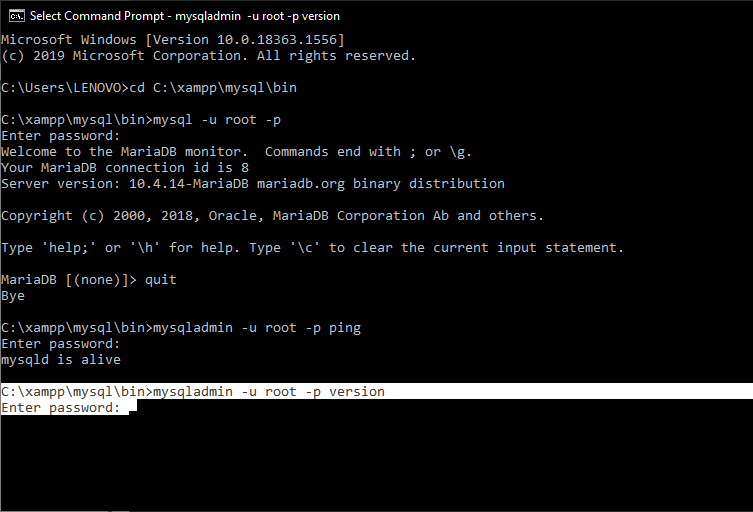
Tampilan MySQL



Mengecek MySQL jalan atau tidak

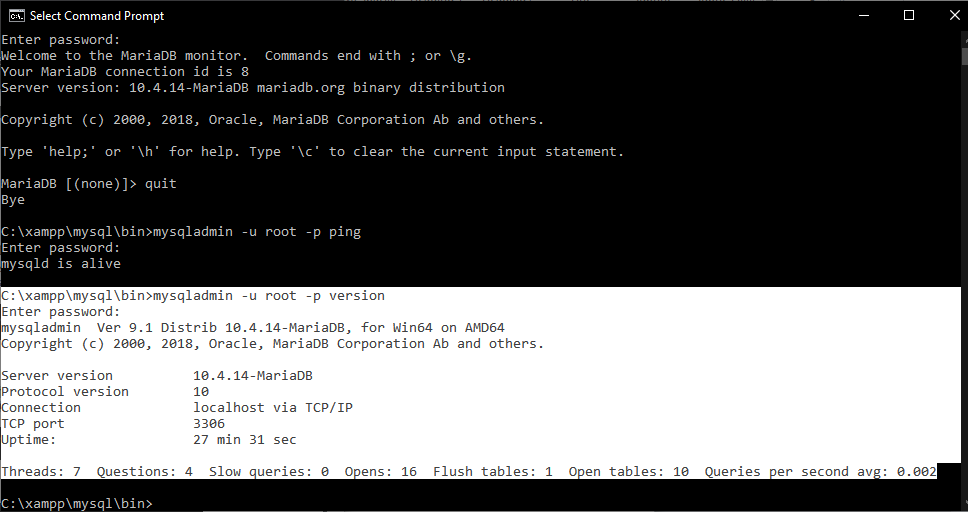


Tampilan Hasil Pengecekan

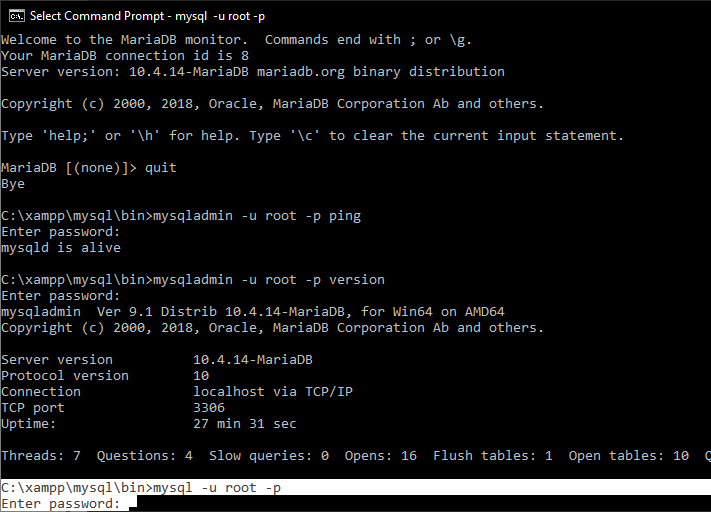


Mengecek versi MySQL yang digunakan menggunakan perintah;

Pastikan sudah keluar dari MySQL sebelum mengecek versinya.

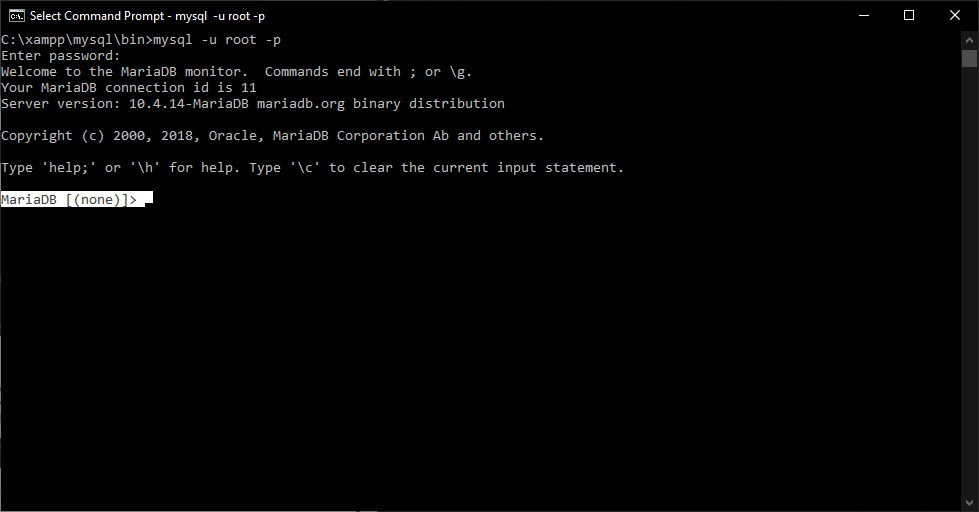


Tampilan hasil pengecekan versi MySQL.

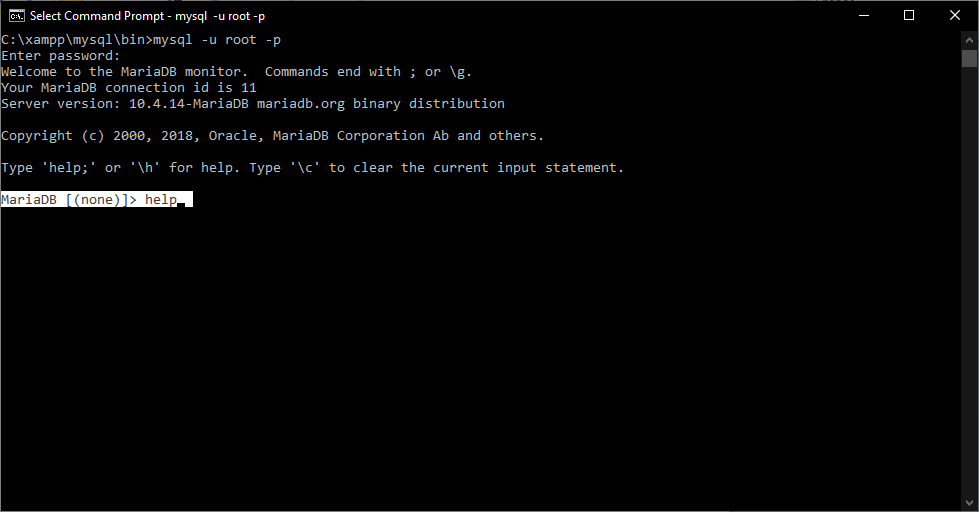


Menampilkan Help MYSQL

Untuk melihat bantuan MySQL, anda harus masuk dulu ke MySQL dengan perintah;



Tanda jika anda sudah ada dalam MySQL adalah;



Untuk menampilkan HELP bisa menggunakan perintah.



Hasil tampilan HELP.



Menampilkan Tanggal

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



Menampilkan Jam atau Waktu



Menggunakan Kalkulator

Untuk menggunakan kalkulator di MySQL gunakan perintah sebagai berikut; Penjumlahan.



Perkalian



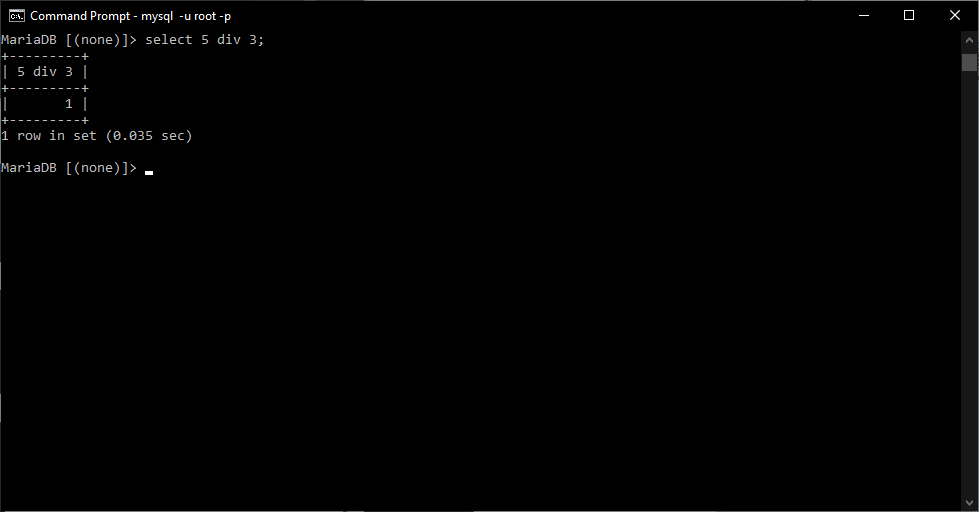
Pengurangan



Pembagian

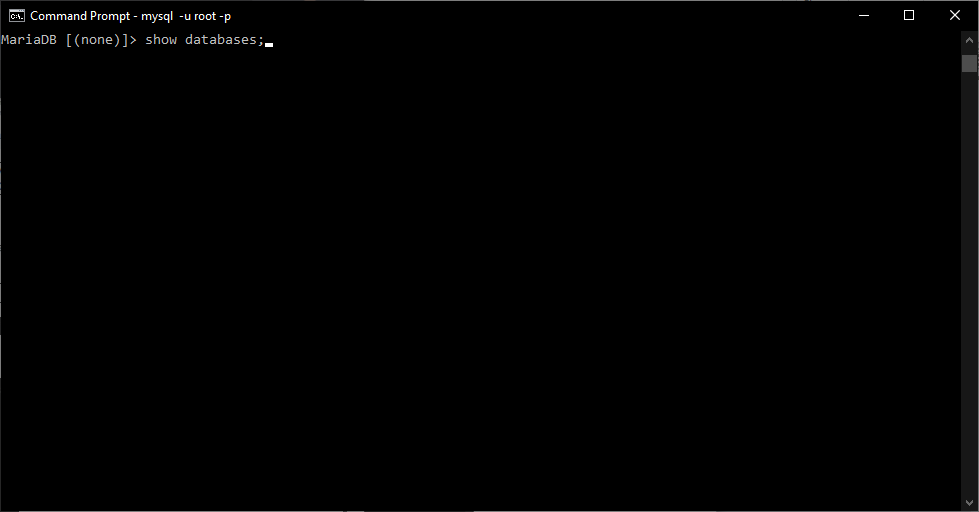


Modulo (sisa hasil pembagian)



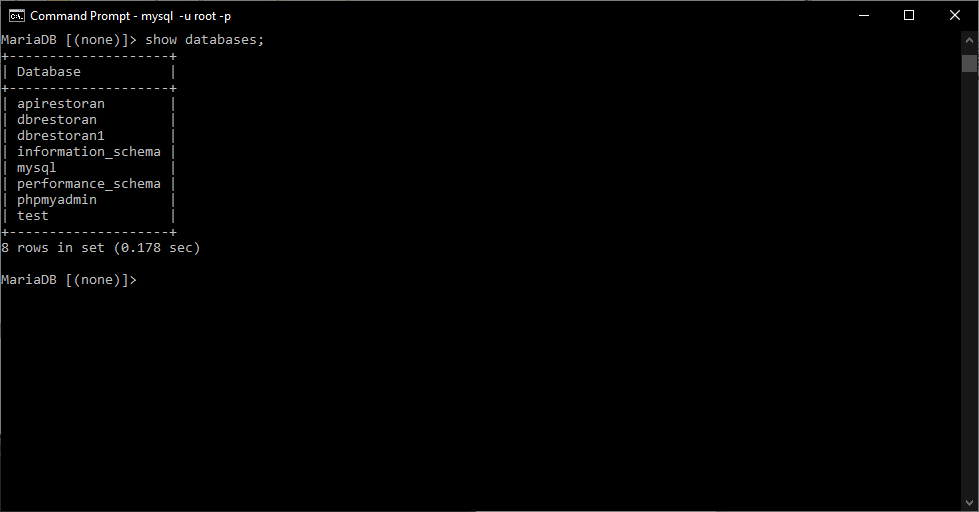
Hasil Pembagian Integer (utuh)

**DDL**

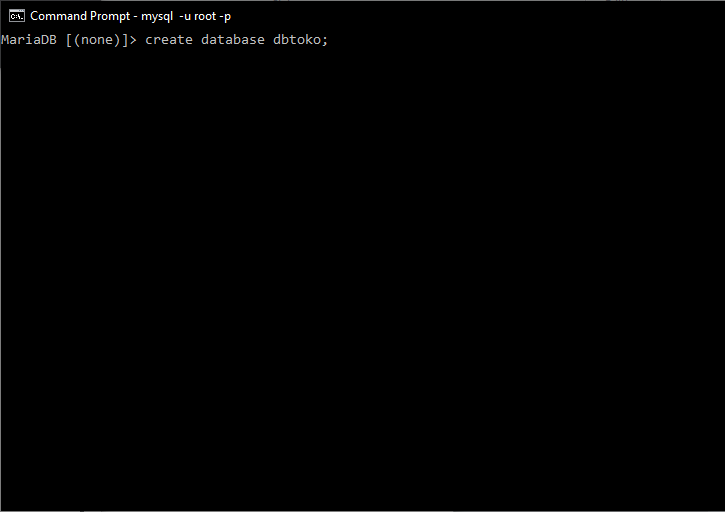


Menampilkan Database

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



Hasil perintah diatas adalah;

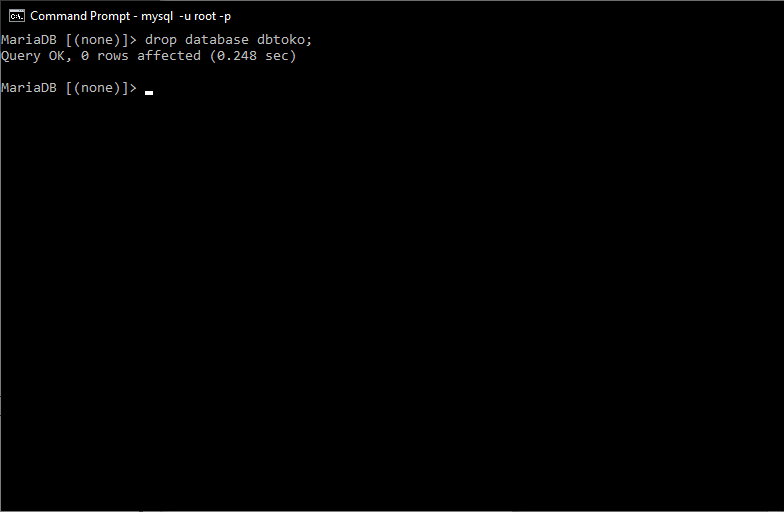


Membuat Database

Pada latihan ini, akan membuat database dengan nama dbtoko. Gunakan perintah berikut;

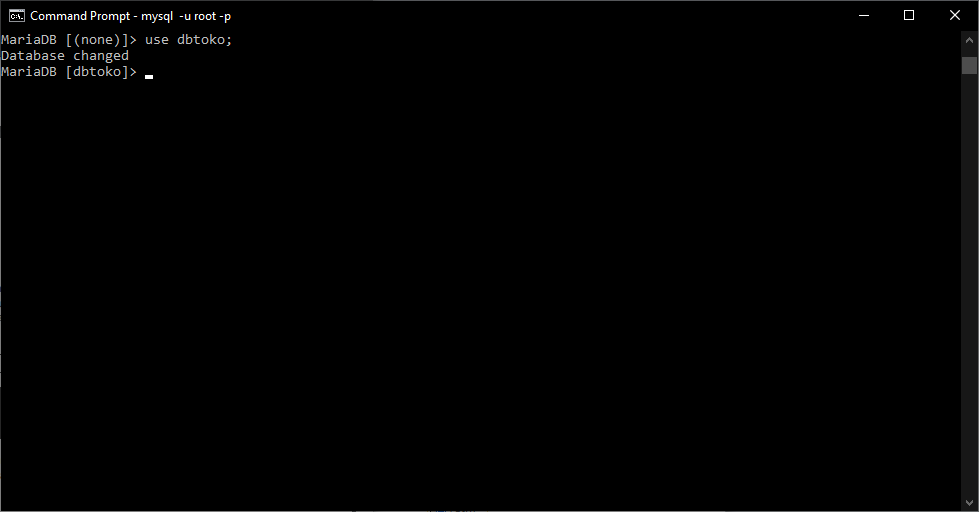


Periksa database yang sudah dibuat dengan menggunakan perintah;



Menghapus Database

Jika ada database yang tidak digunakan bisa dihapus dengan perintah;

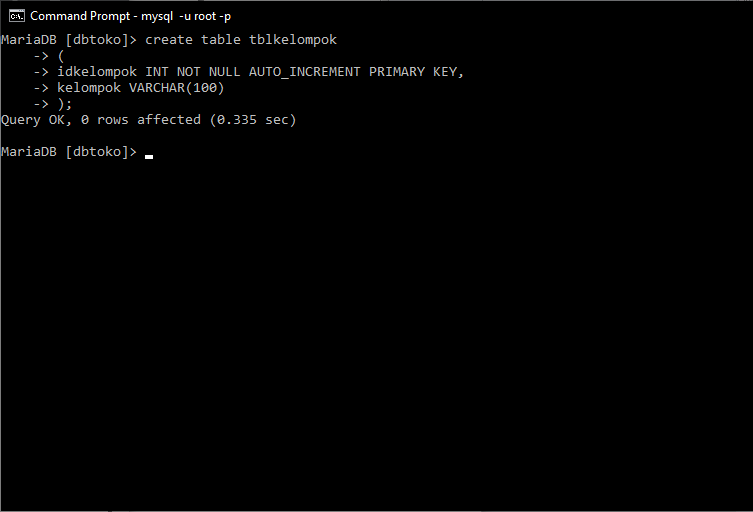


Mengaktifkan Database

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] .



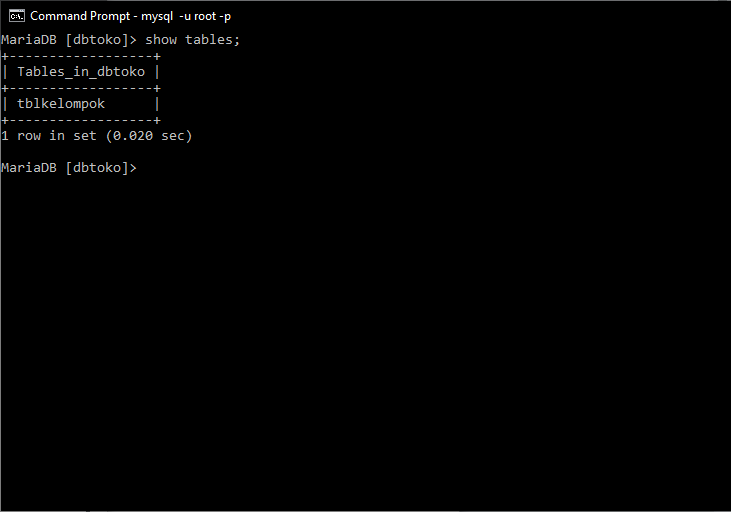
MEMBUAT TABEL

Pembuatan tabel berdasarkan desain yang sudah dibuat sebelumnya, tabel yang pertama dibuat adalah tabel tblkelompok dengan desain sebagai berikut;

Pastikan sebelum membuat tabel database sudah dipilih dengan benar; gunakan perintah berikut untuk pembuatan tabel. Pada desain yang dibuat menggunakan MS Access tipe data yang digunakan adalah text maka pada MySQL menggunakan VARCHAR

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

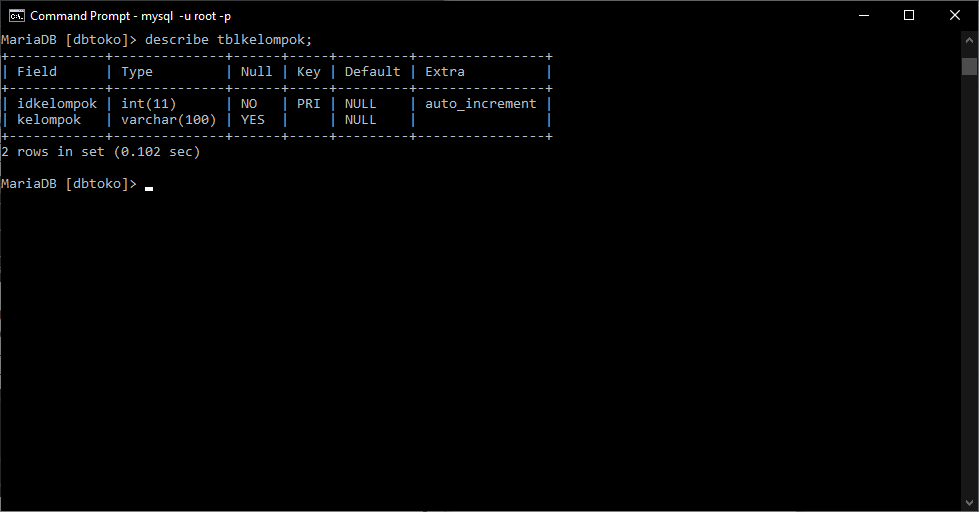
**Untuk berpindah baris gunakan ENTER.**



MENAMPILKAN TABEL

Untuk menampilkan tabel yang sudah dibuat, gunakan perintah berikut;

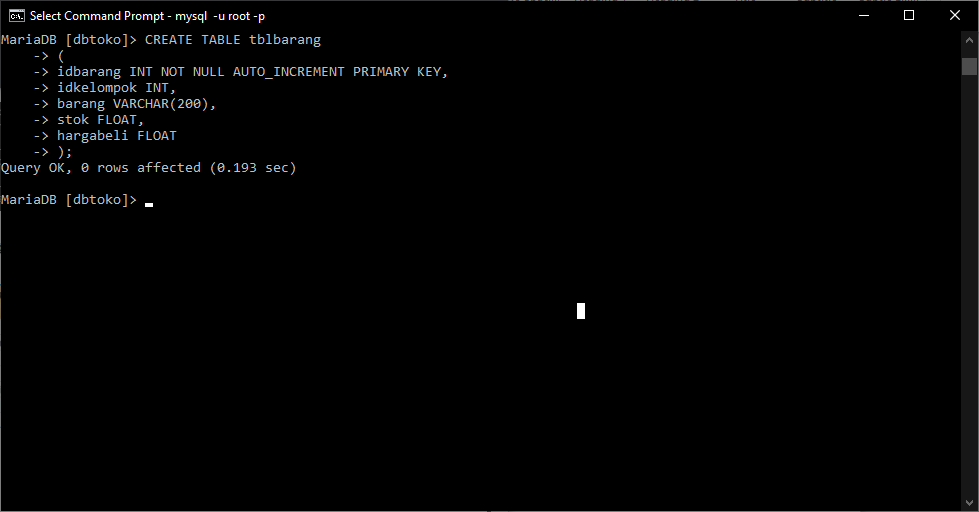
**SHOW TABLES;**



MENAMPILKAN STRUKTUR TABEL

Untuk melihat struktur tabel gunakan perintah sebagai berikut;

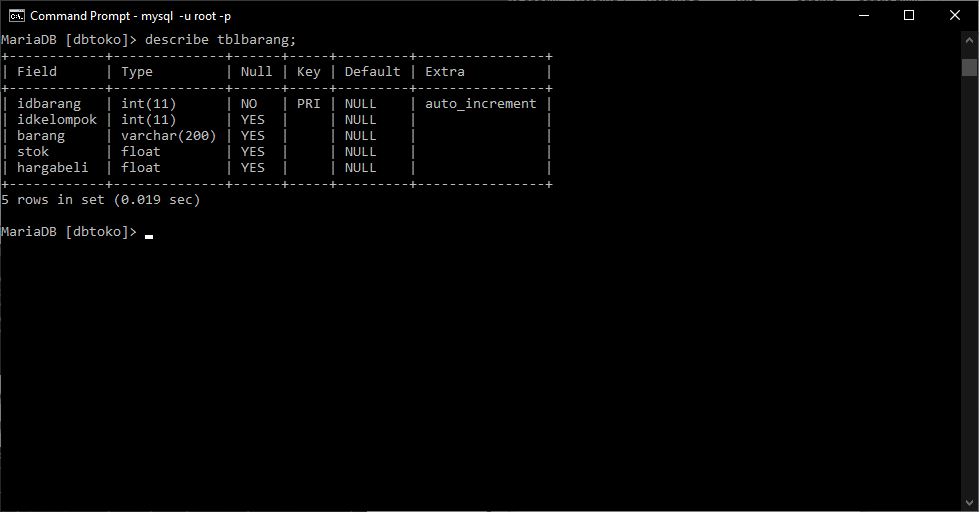
**DESCRIBE nama\_tabel;**



Untuk membuat tabel tblbarang gunakan perintah sebagai berikut;

Karena pada MySQL tidak ada tipe data Currency maka kita bisa menggunakan tipe data FLOAT

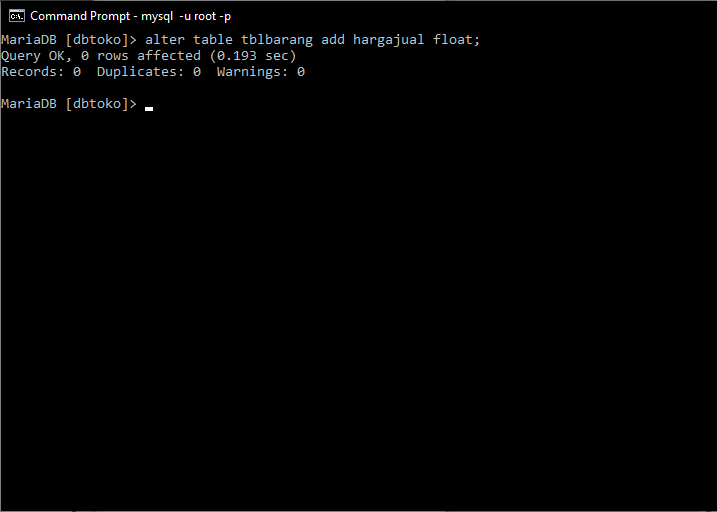
Periksa apakah struktur yang dibuat sudah benar





Menampilkan Tabel

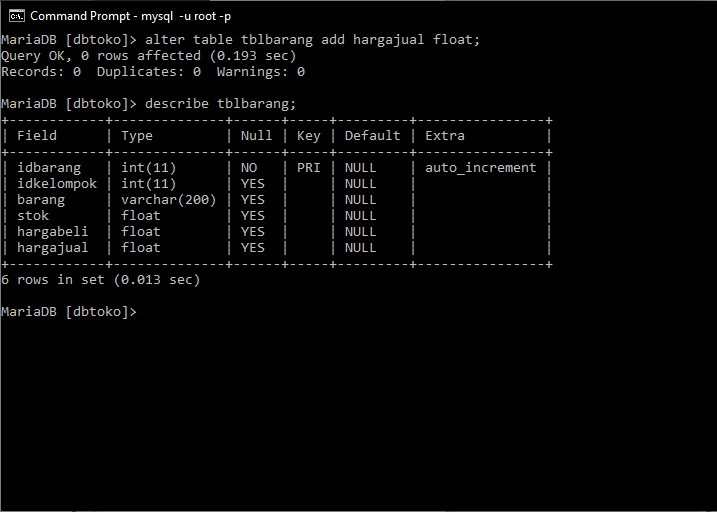
Tabel yang sudah dibuat pada database bisa dilihat dengan menggunakan perintah berkut;



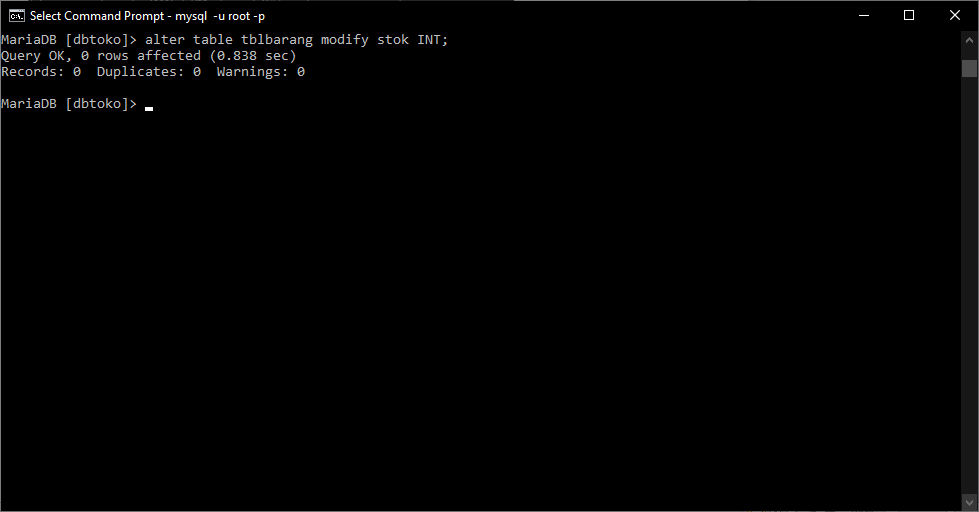
MENAMBAH KOLOM

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;**



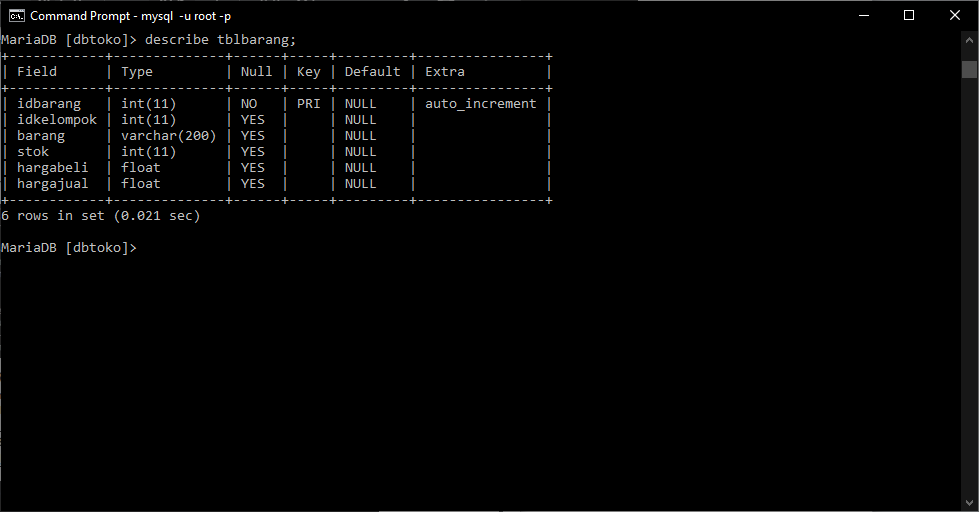
Setelah penambahan kolom periksa tblbarang dengan perintah sebagai berikut;



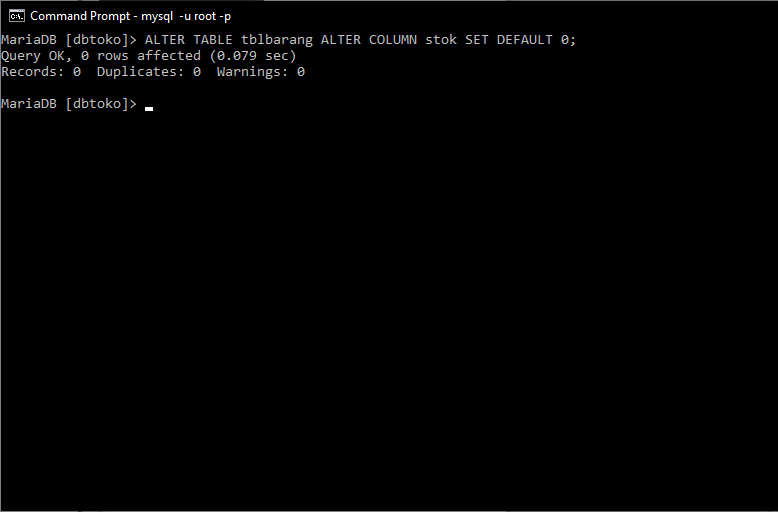
MERUBAH TIPE DATA

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;**



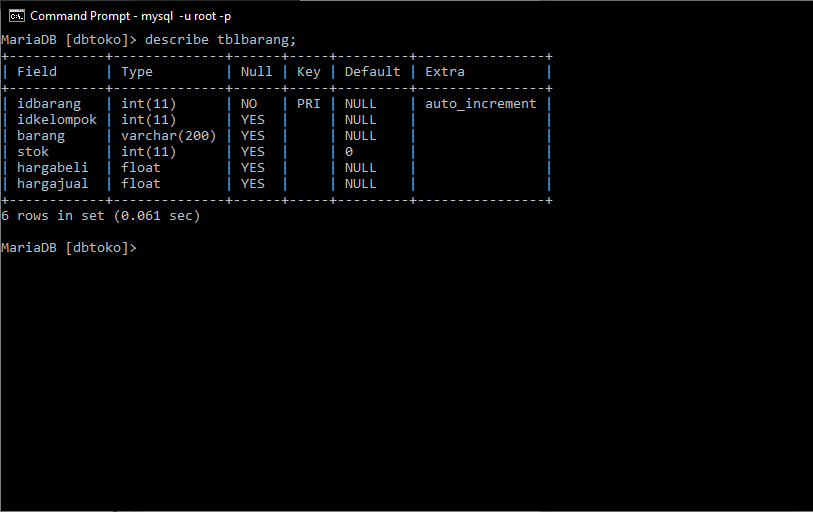
Periksa kembali tipe data yang digunakan menggunakan perintah berikut;



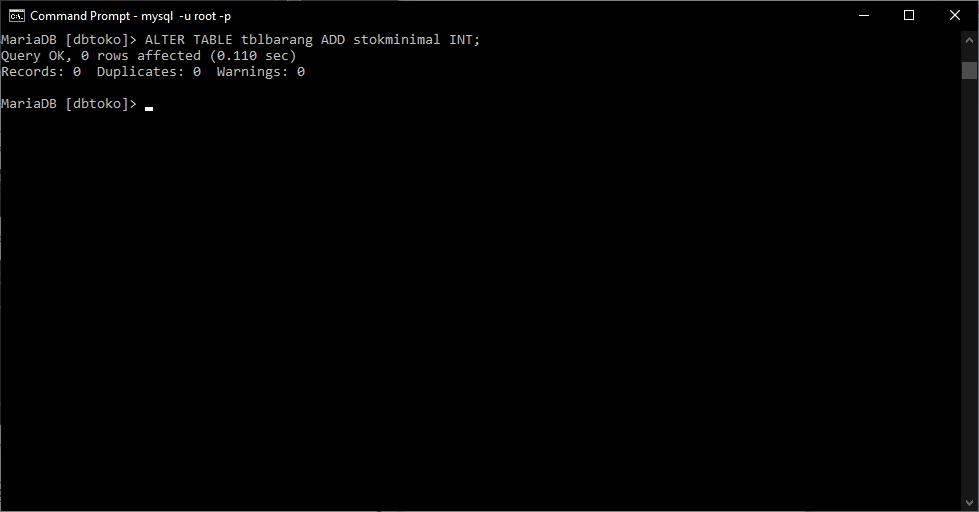
MEMBERI NILAI DEFAULT PADA KOLOM

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;**

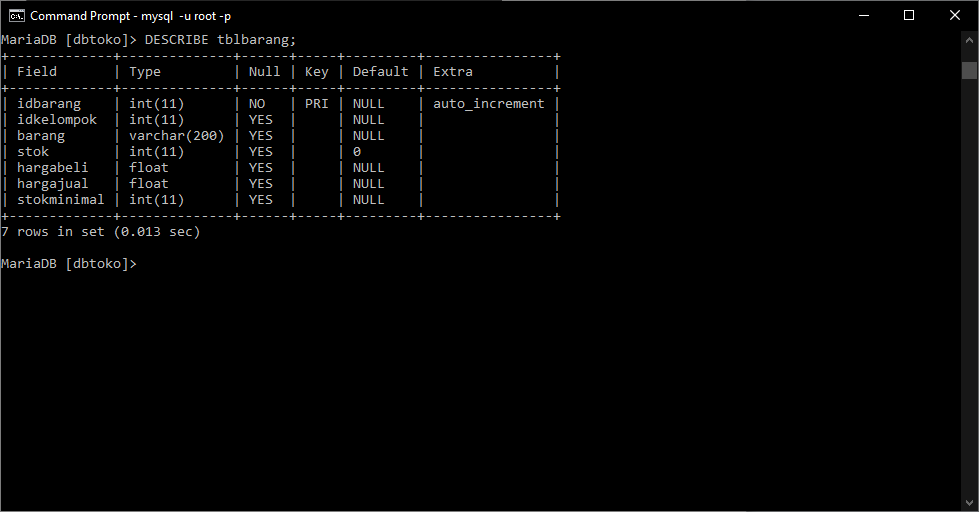


Periksa dengan menggunakan perintah berikut;

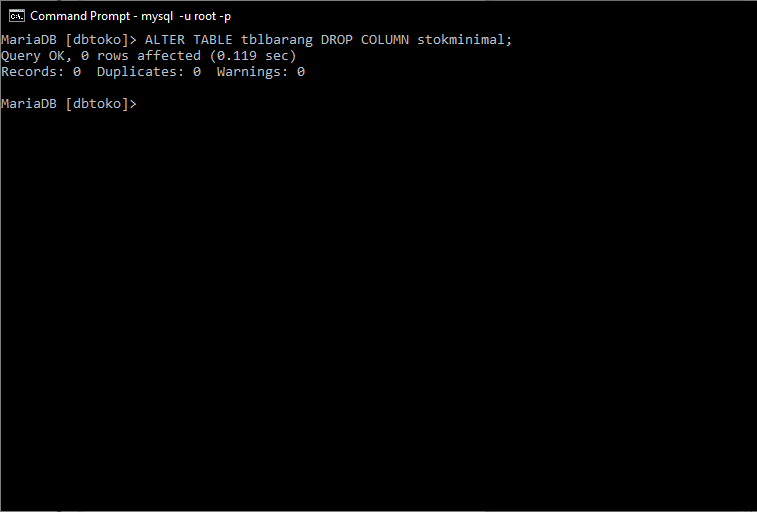


MENGHAPUS KOLOM

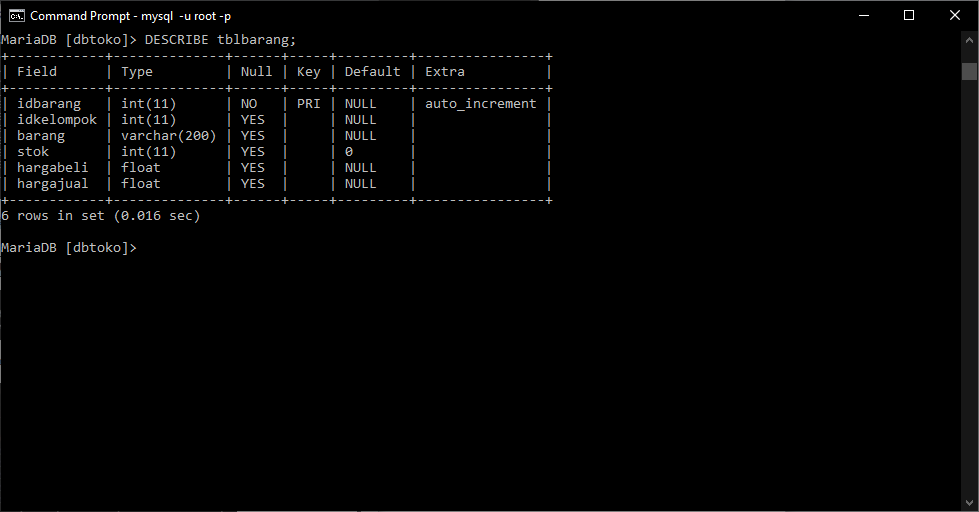
Pada latihan untuk menghapus kolom, tambahkan kolom baru terlebih dahulu dengan menggunakan perintah sebagai berikut;



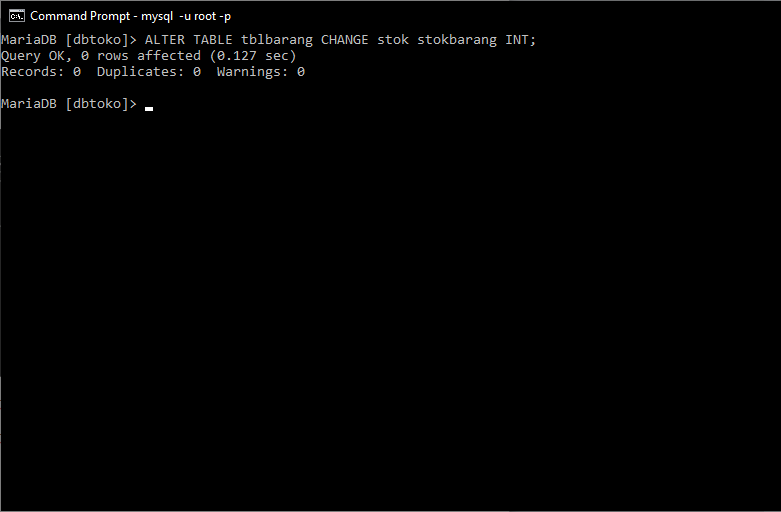
Periksa struktur tabel yang sudah dibuat;



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



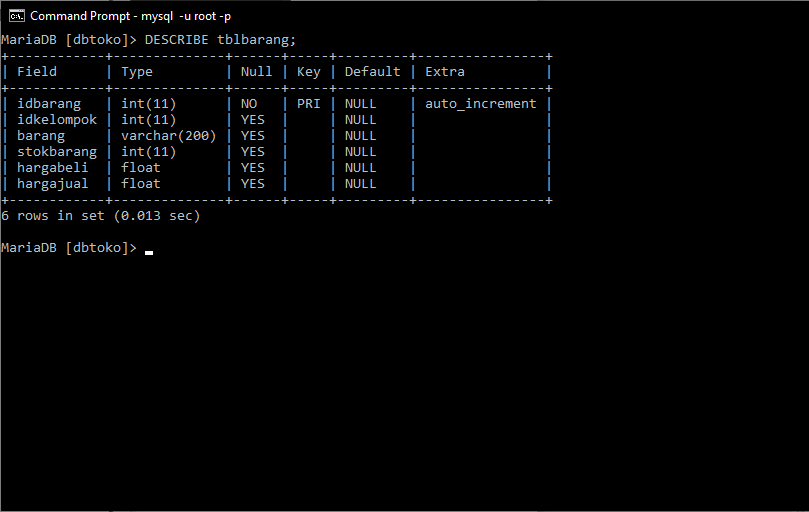
Periksa struktur tabel dengan perintah berikut;



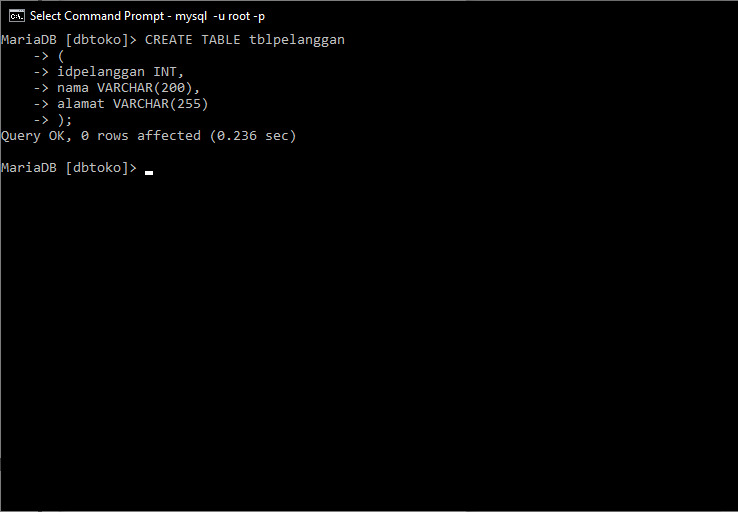
MERUBAH NAMA KOLOM

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

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

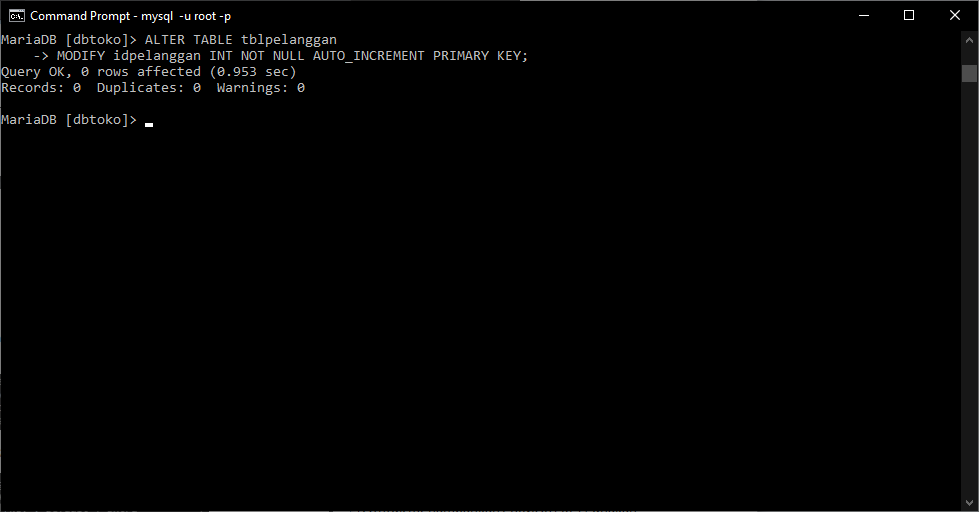


Periksa struktur tabelnya;



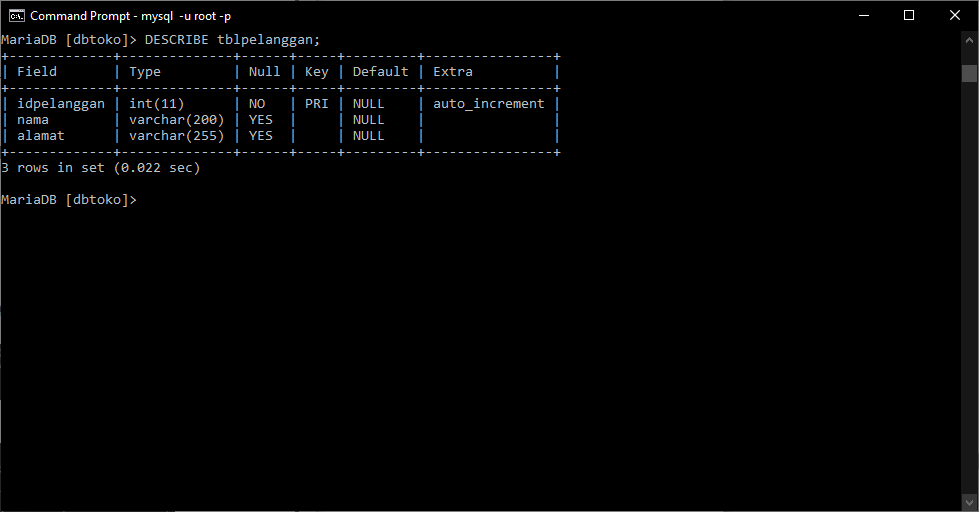
MEMBUAT TABEL tblpelanggan

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

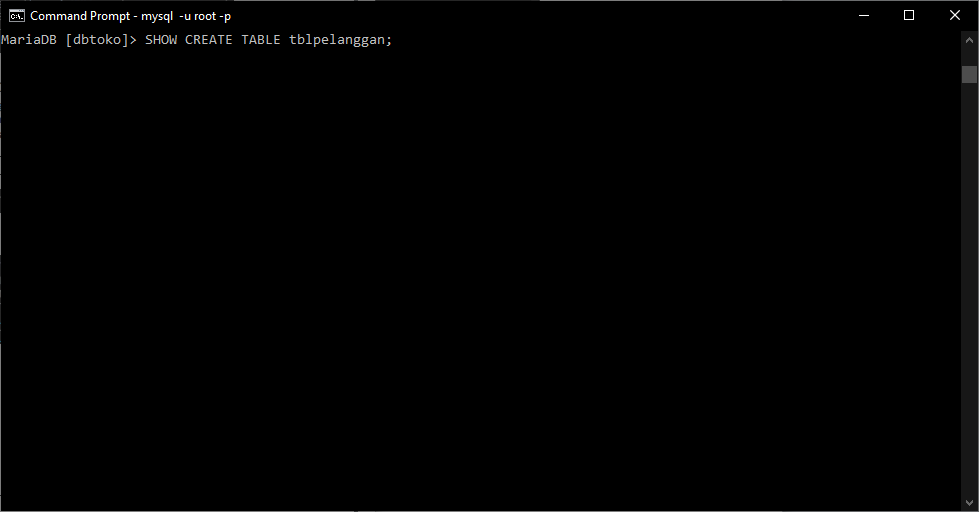


MENAMBAHKAN PRIMARY KEY

Perbaikan pada pembuatan tabel diatas dengan merubah tipe data pada kolom dan menambahkan PRIMARY KEY;



Periksa struktur tabel dengan perintah berikut;



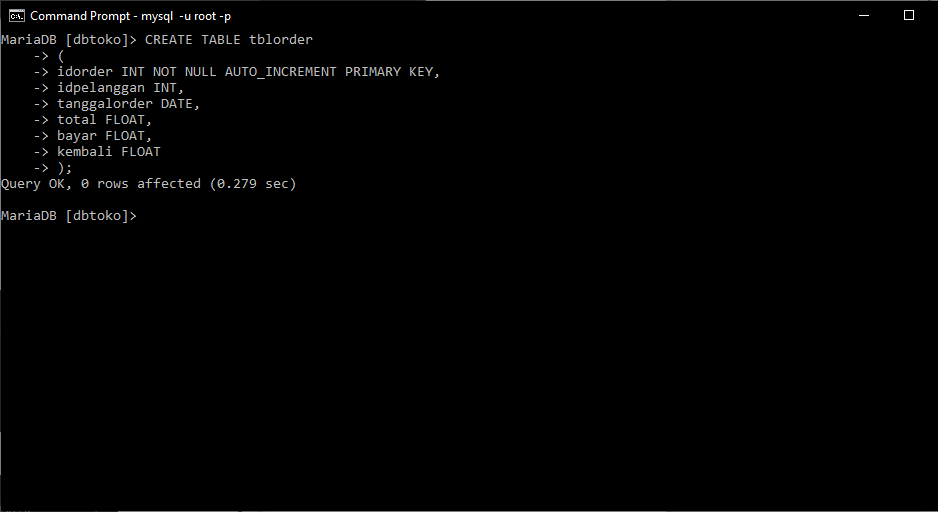
MENAMPILKAN ENGINE YANG DIGUNAKAN

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;**

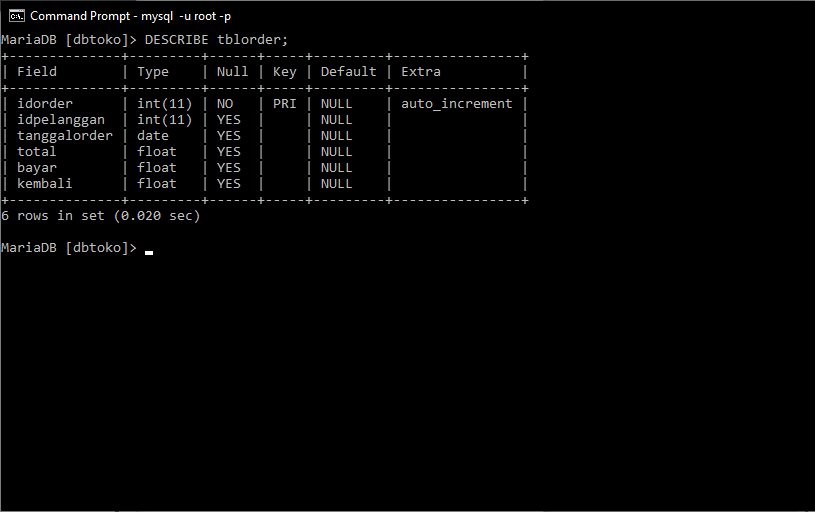


Hasil yang ditampilkan adalah;



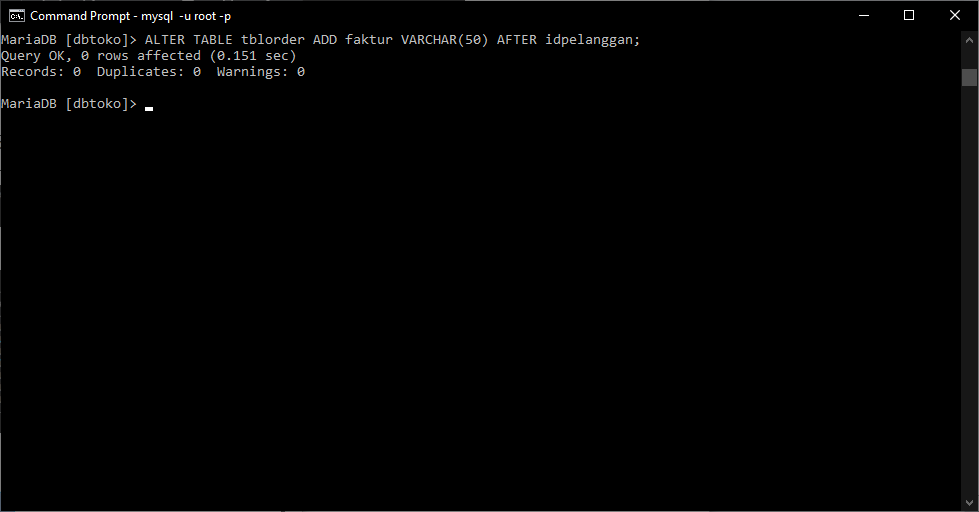
MEMBUAT TABEL tblorder

Perintah untuk membuat tblorder;



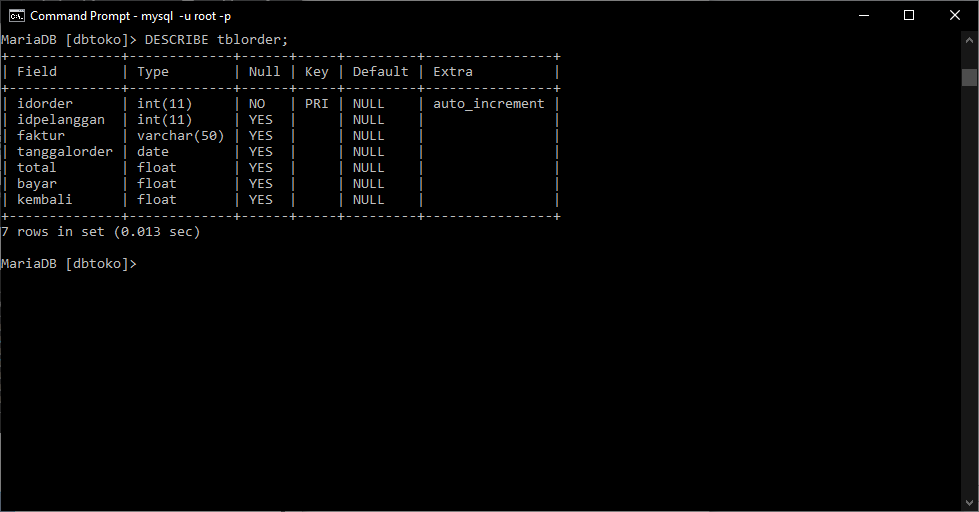
Struktur tabel

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

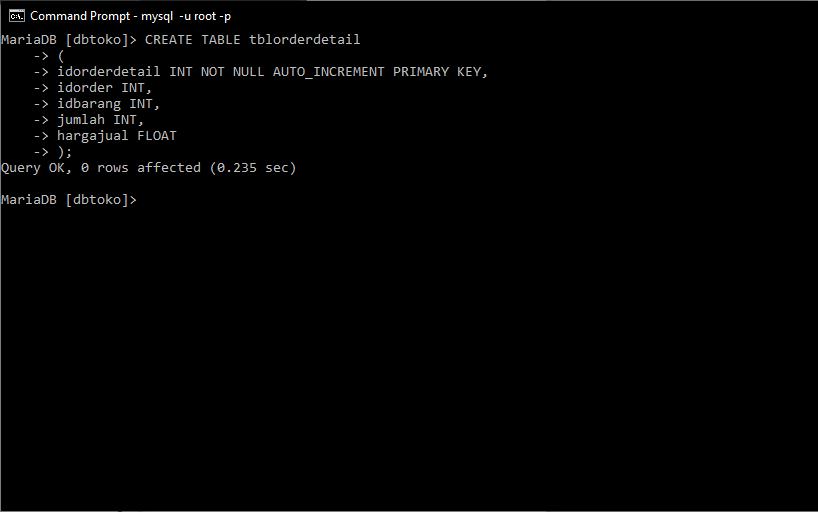


MENAMBAH KOLOM SETELAH KOLOM

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

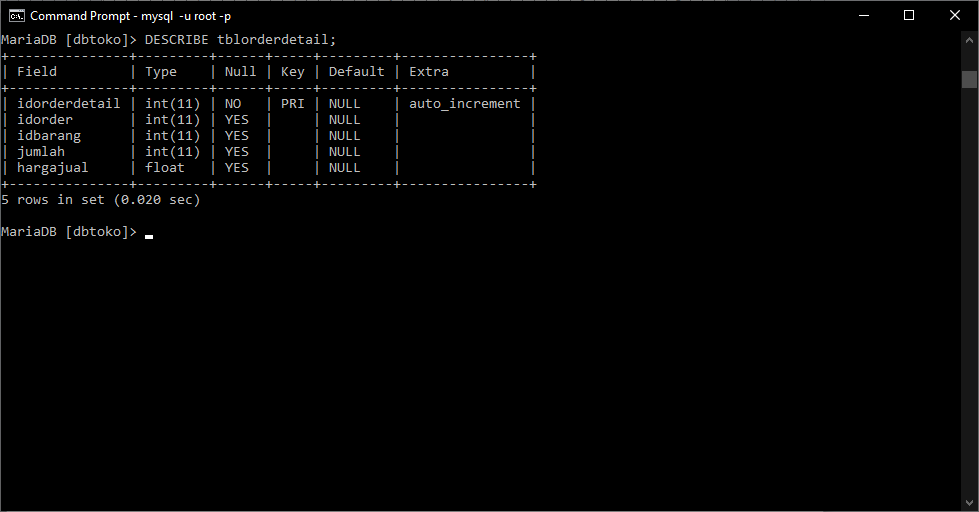


Hasilnya adalah;

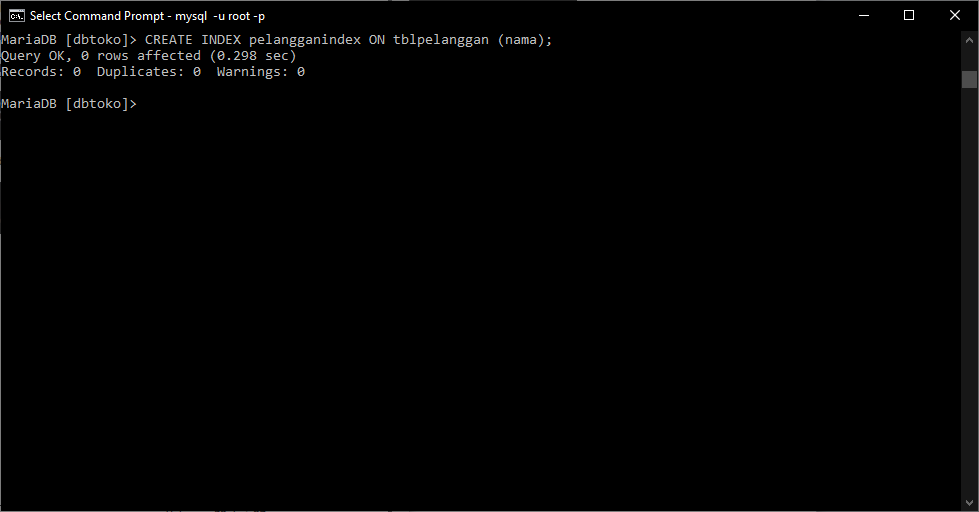


MEMBUAT TABEL tblorderdetail

Perintah membuat tblorderdetail



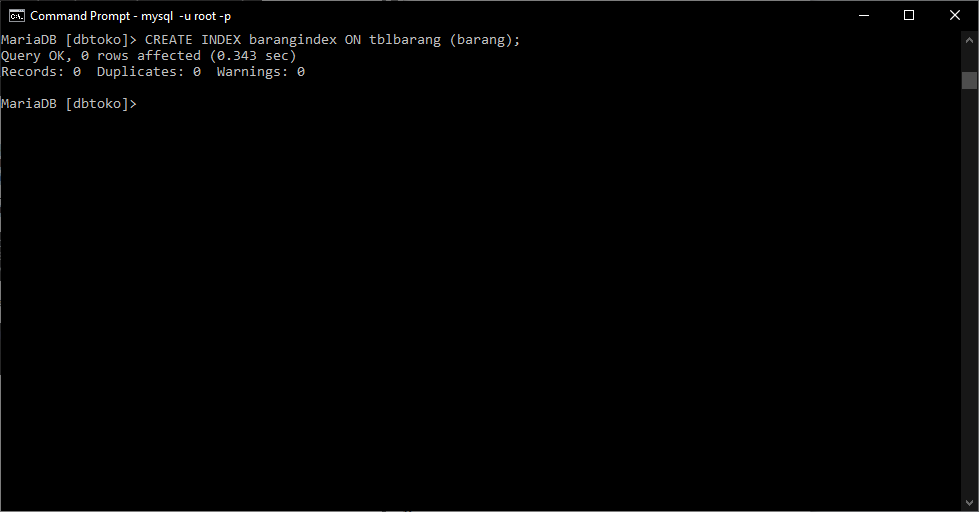
Struktur tabel yang sudah dibuat;



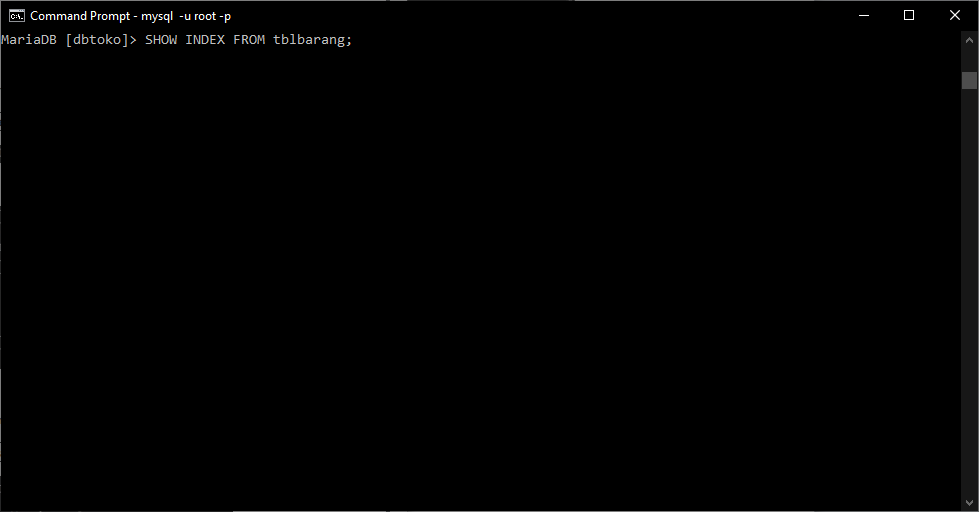
MEMBUAT INDEX

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);**



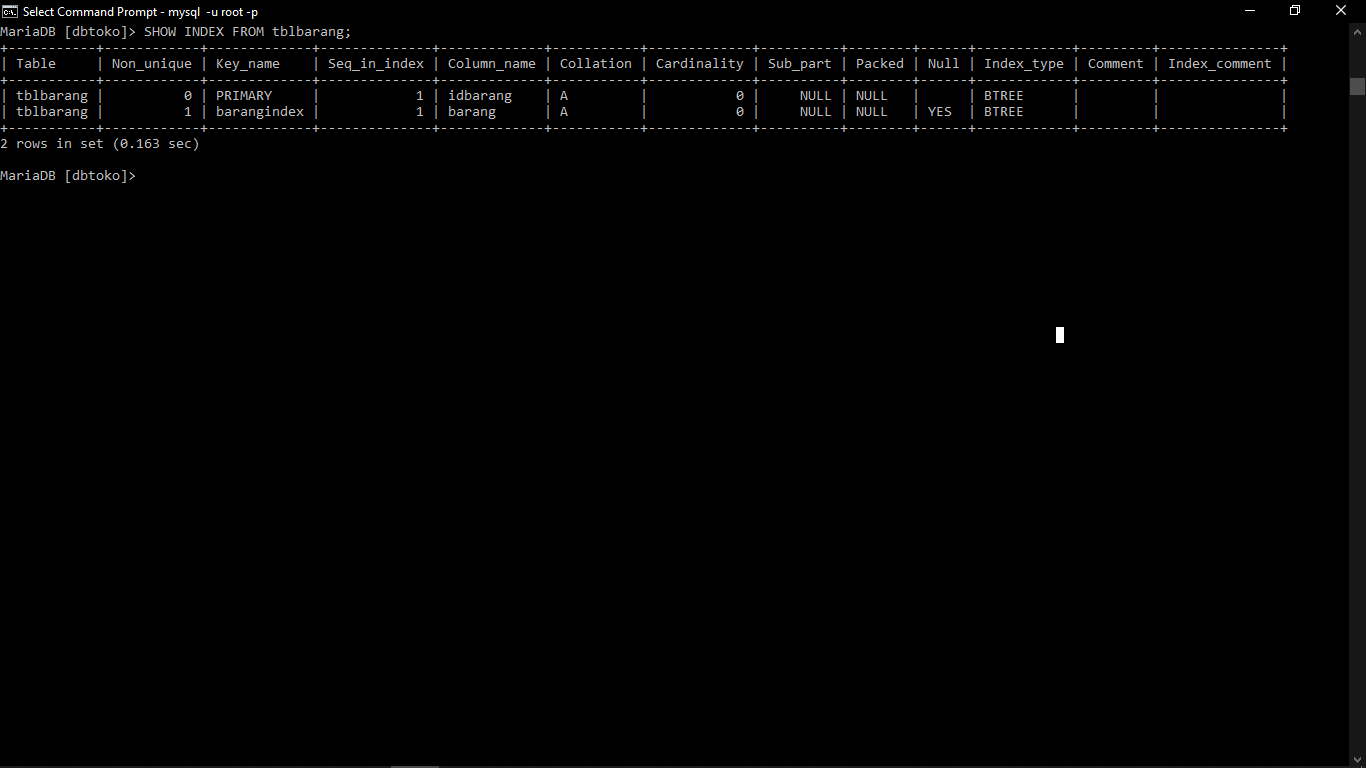
Buat index untuk tabel tblbarang dengan perintah sebagai berikut;



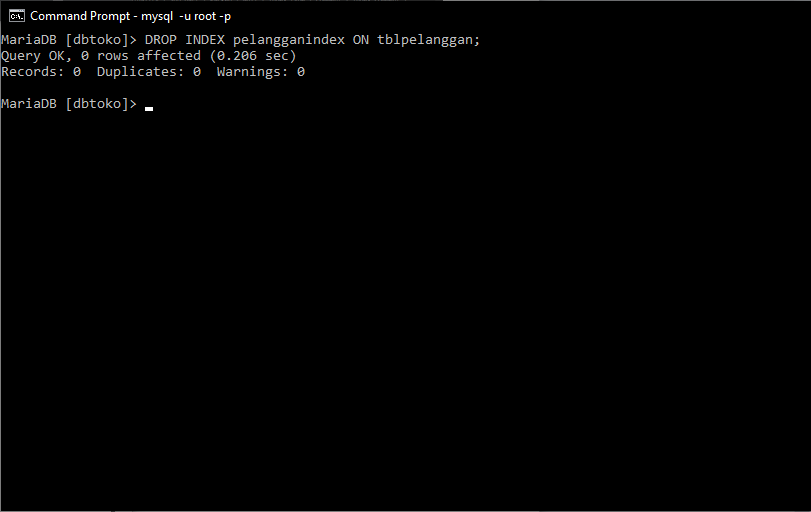
MENAMPILKAN INDEX

Index yang sudah dibuat bisa ditampilkan dengan perintah;

**SHOW INDEX FROM nama\_tabel;**



Hasilnya adalah;

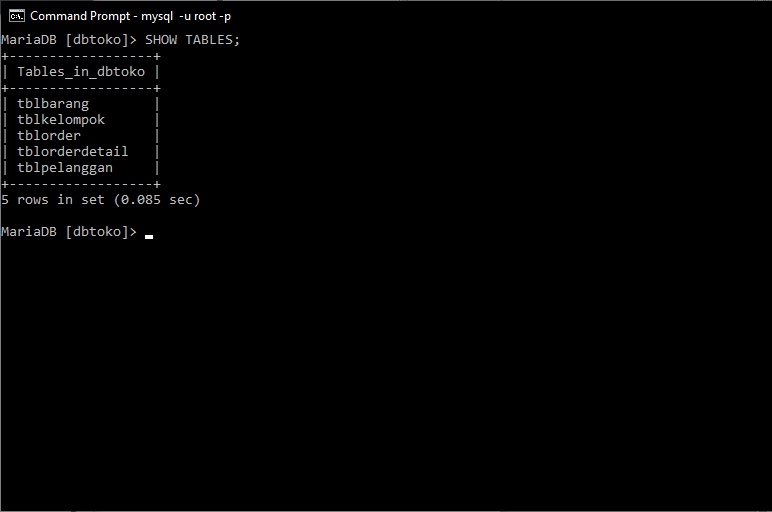


MENGHAPUS INDEX

Jika INDEX tidak digunakan bisa dihapus dengan perintah;

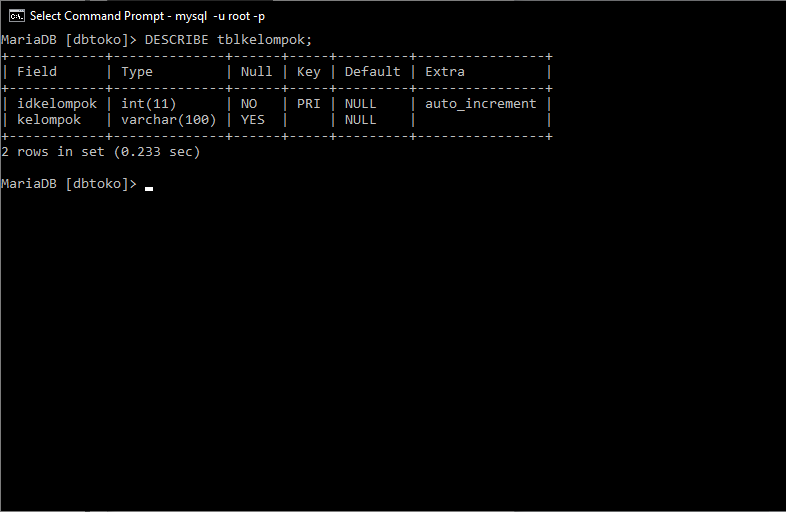
**DROP INDEX nama\_index ON nama\_tabel;**

**DML**

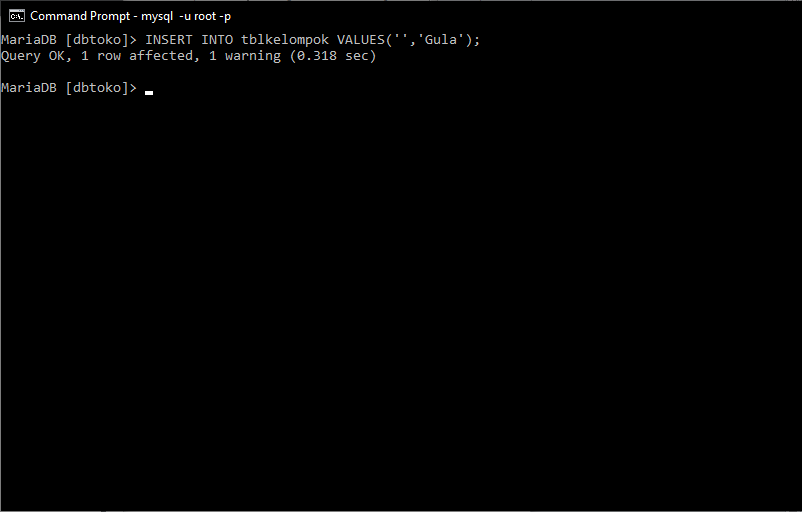


INSERT SEMUA KOLOM

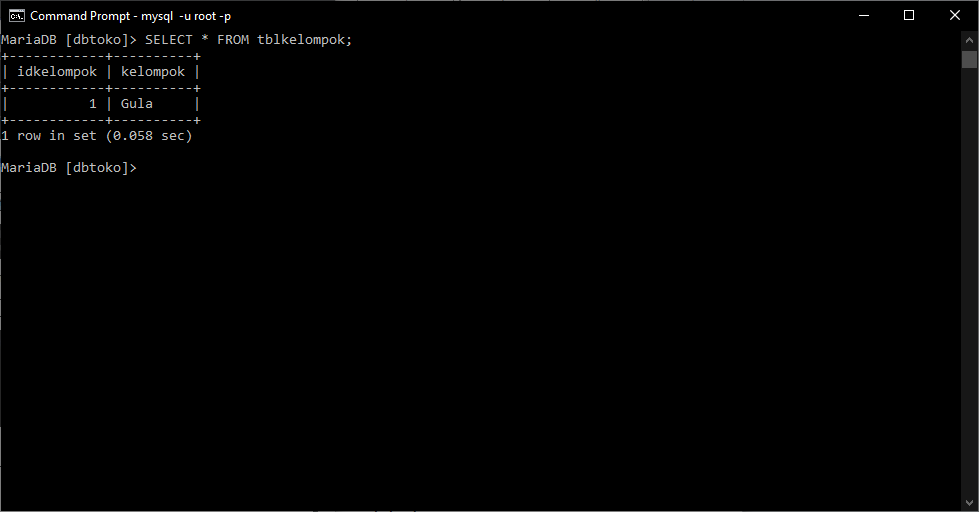
Insert semua kolom digunakan untuk memasukan data pada semua kolom tabel. Sebelum melakukan INSERT data periksa dulu tabel yang akan digunakan.



Sebagai latihan akan menggunakan tblkelompok, lihat dulu kolom tblkelompok

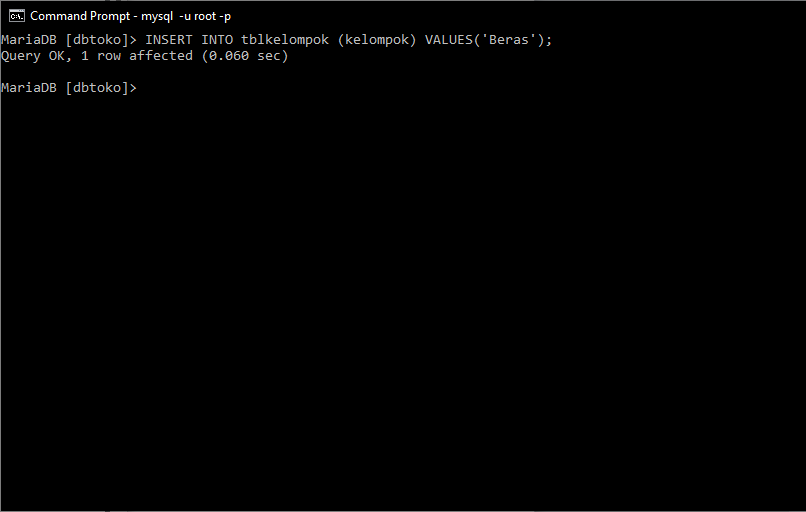


Pada tblkelompok terdapat 2 kolom yang akan di INSERT datanya, yaitu [idkelompok], [kelompok]. **INSERT INTO nama\_tabel VALUES (kolom, kolom, semua\_kolom);**



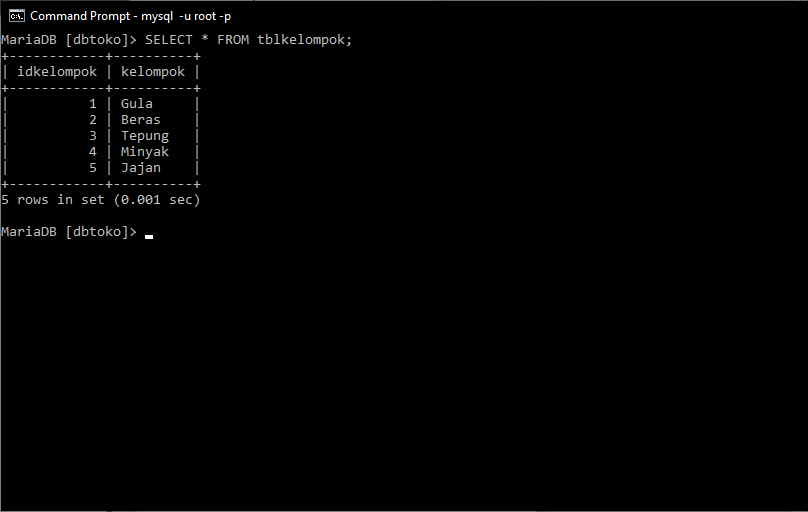
Untuk memeriksa apakah data yang di INSERT kan berhasil gunakan perintah;

**SELECT \* FROM nama\_tabel;**



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);**



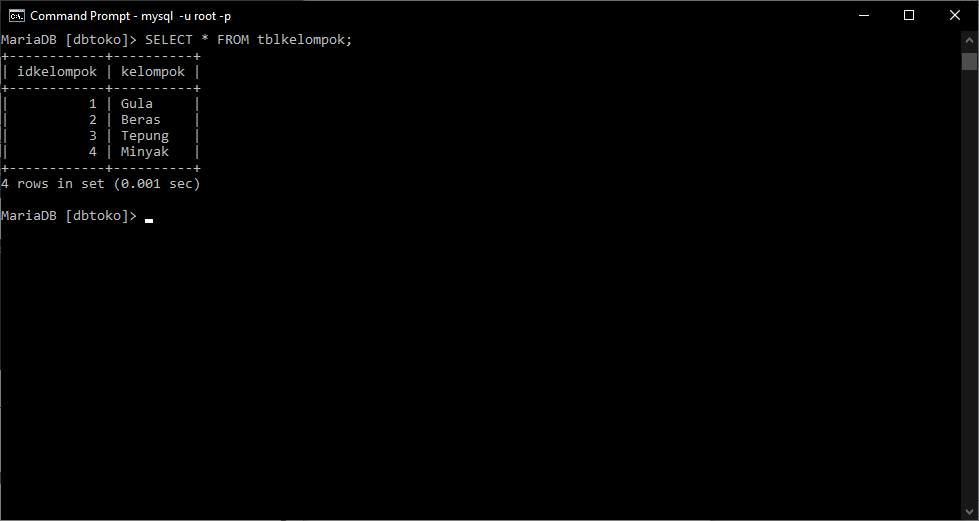
Sebagai latihan lakukan INSERT data sehingga diperoleh data sebagai berikut;



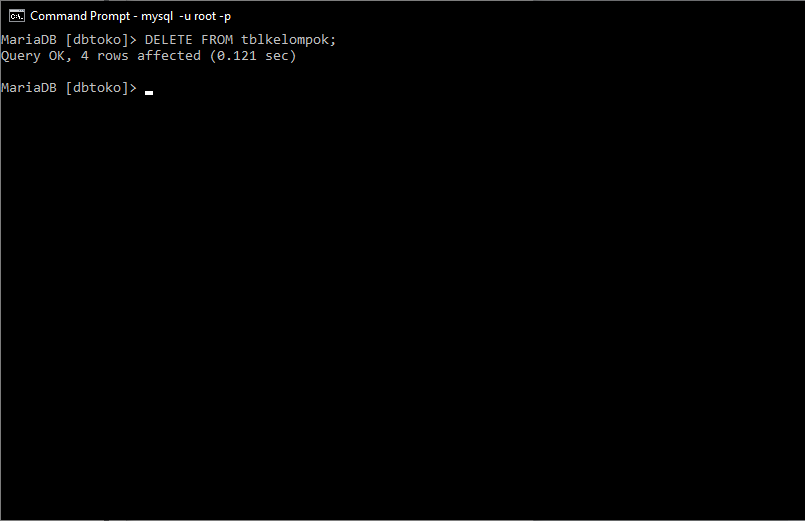
DELETE SEBAGIAN RECORD (BARIS DATA)

Untuk menghapus sebagian baris gunakan perintah;

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



Baris Data (Record) setelah ada yang dihapus.



DELETE SEMUA RECORD

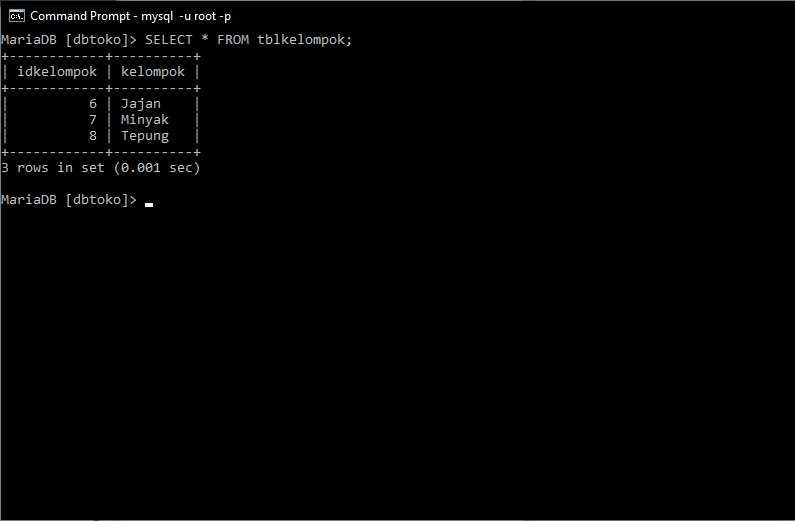
Untuk menghapus semua baris data bisa dilakukan dengan perintah;

**DELETE FROM nama\_tabel;**



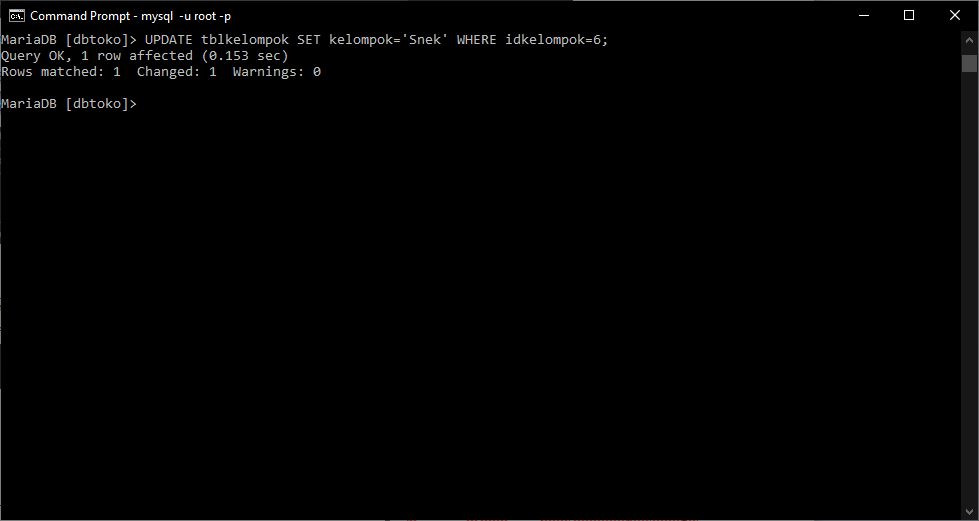
Periksa tabel yang sudah dihapus menggunakan perintah;

Karena data sudah dihapus maka yang tampil adalah KOSONG.



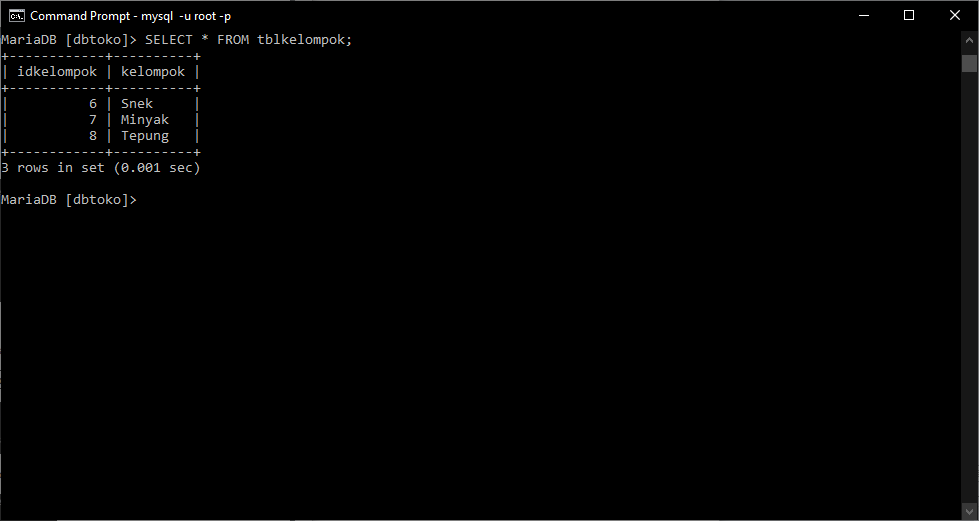
UPDATE SEBAGIAN RECORD

Karena semua data sudah dihapus semua pada materi delete, isi dulu data pada tblkelompok sebagai latihan dengan data sebagai berikut;

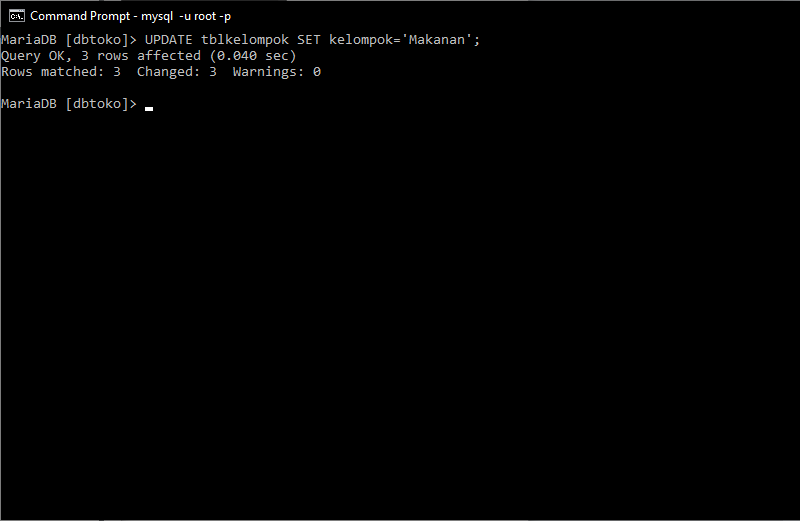


Pada tabel diatas UBAH data ‘Jajan’ menjadi ‘Snek’ menggunakan perintah sebagai berikut;

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



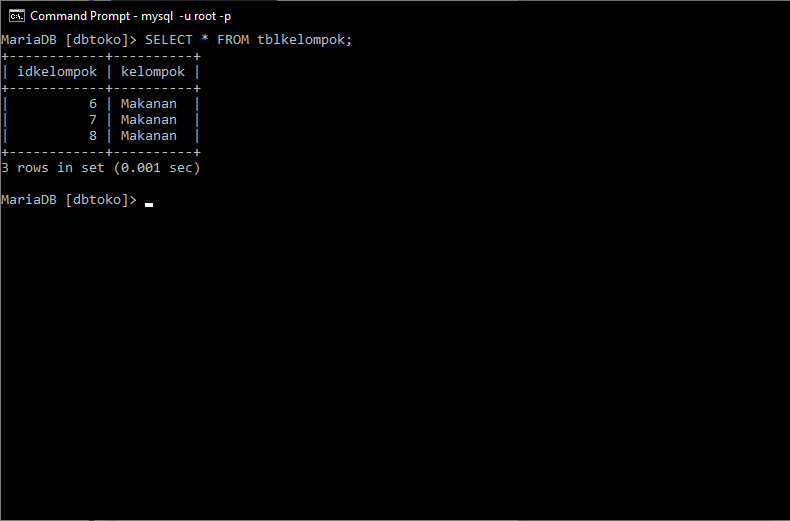
Periksa data yang sudah diubah



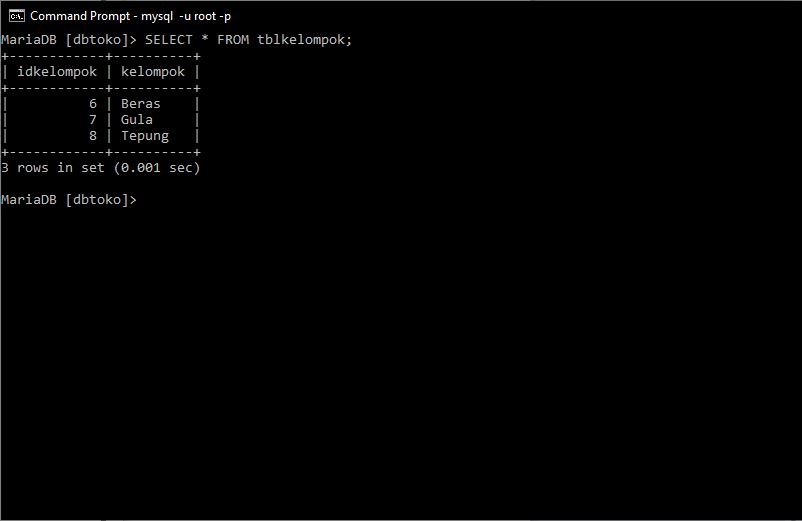
UPDATE SEMUA RECORD

Untuk mengubah semua data bisa dilakukan dengan perintah sebagai berikut;

**UPDATE nama\_tabel SET nama\_kolom=isi\_kolom;**



Periksa data yang sudah diubah dengan perintah berikut;



Lakukan UPDATE lagi sebagai latihan sehingga hasilnya sebagai berikut;

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.

RELATIONAL 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).

PROSES BISNIS (ALUR KERJA)

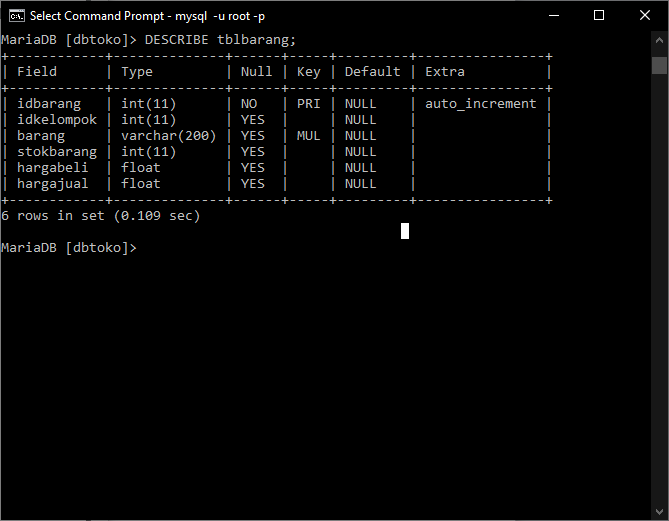
Pada relasi tersebut terdapat alur kerja yang sering disebut dengan PROSES BISNIS. Proses bisnis pada relasi tersebut yaitu: Sebelum toko dibuka pemilik akan mengelompokan barang – barang yang akan dijual. Kelompok barang tersebut akan disimpan di tabel kelompok.

Contoh: Toko akan menjual Beras, Gula, Tepung

Kemudian pada kelompok tersebut akan dibuat DETAIL isinya.

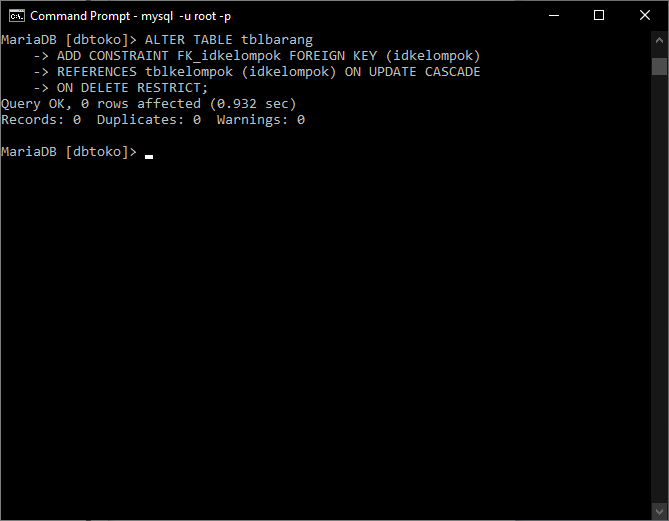
Pada pembuatan database setiap kelompok akan di wakili oleh kolom PRIMARY KEY yang akan dimasukan ke tabel yang lain untuk menunjukan bahwa kolom tersebut adalah anggota dari tabel yang lain.

Pada tabel [tblbarang] diatas, [idkelompok] akan menunjukan bahwa kolom tersebut adalah isi dari tabel [tblkelompok]



MEMBUAT RELASI ANTAR TABEL

Lihat dulu struktur tabel yang akan dibat relasinya. Pada tblbarang terlihat struktur seperti pada gambar

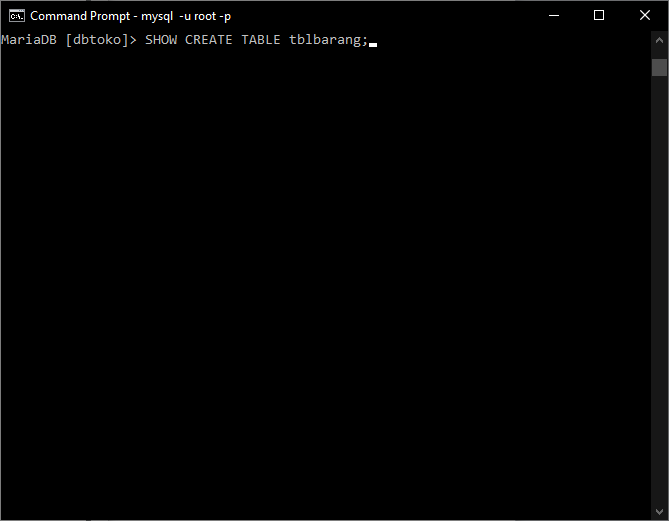


Perintah untuk membuat relasi adalah sebagai berikut;

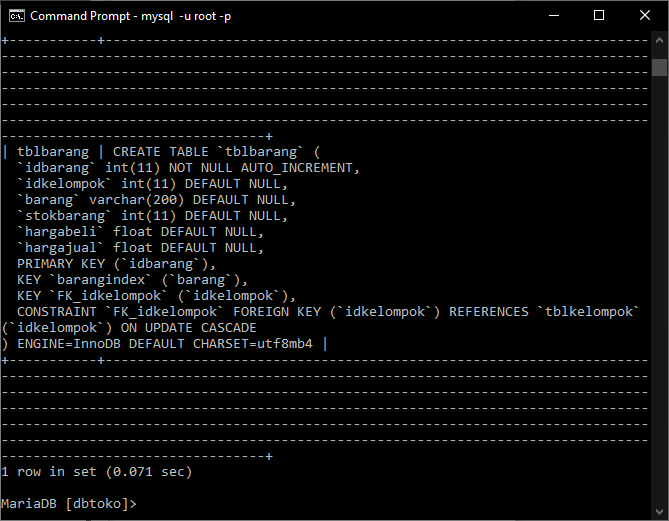
**ALTER TABLE nama\_tabel\_detail ADD CONSTRAINT FK\_foreign\_key\_pada\_tabel\_detail FOREIGN KEY (foreign\_key) REFRENCES nama\_tabel\_master (primary\_key\_tabel\_master) ON UPDATE CASCADE ON DELETE RESTRICT ;**

ON UPDATE CASCADE : jika kolom primary key pada tabel master maka kolom foreign key pada tabel transaksi akan berubah.

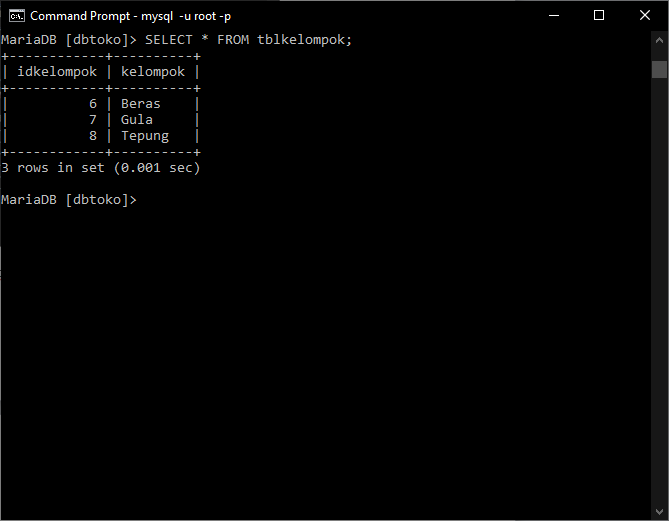
ON DELETE RESTRICT : Kolom primary key di tabel master TIDAK BISA di hapus jika kolom tersebut masih digunakan pada tabel transaksi. ON DELETE CASCADE jika baris pada kolom master dihapus maka semua baris data yang menggunakan isi tabel master pada tabel transaksi akan terhapus :



Untuk melihat hasil pembuatan relasi gunakan perintah berikut;

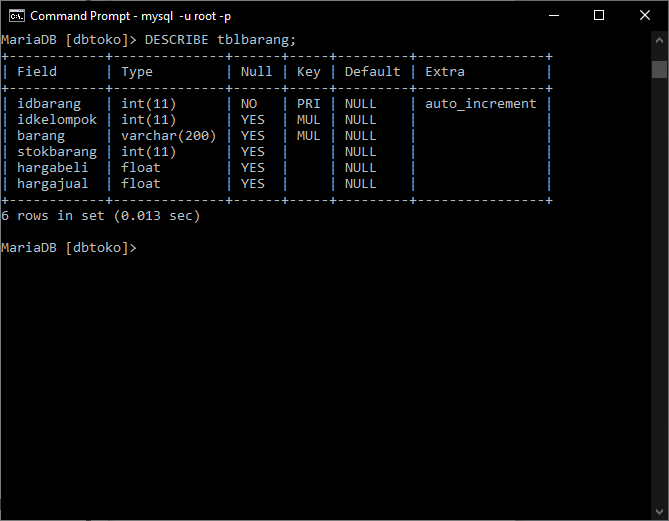


Tampilan jika relasi berhasil dibuat adalah;



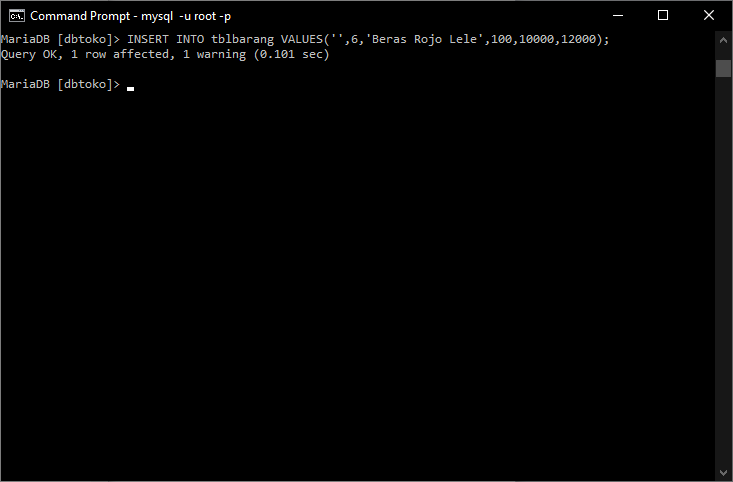
MENGUJI HASIL PEMBUATAN RELASI

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

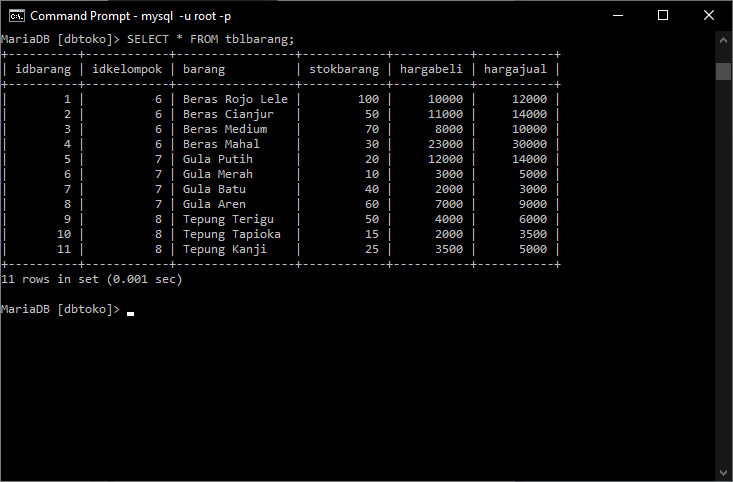


Tampilkan struktur tabel untuk memudahkan proses insert data;

Isi tabel transaksi sesuai dengan idkelompok yang akan digunakan



Karena iDCLrang AUTO\_INCREMENT maka kolom bisa diganti dengan ‘’ (petik kosong)

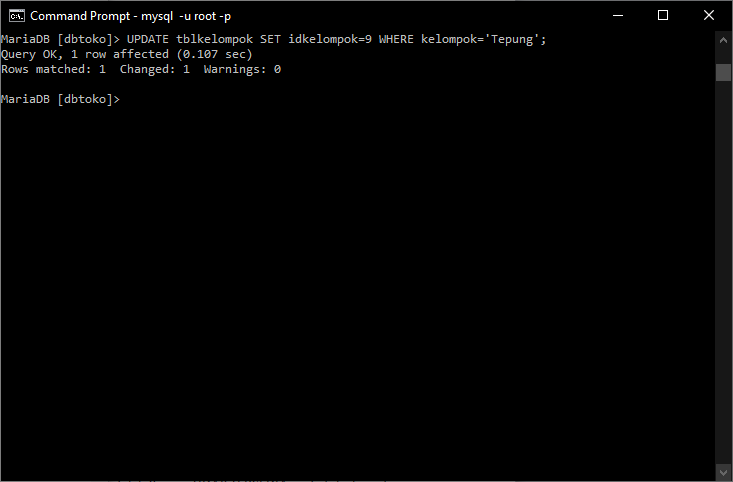


Lakukan insert data sehingga muncul tabel berikut;

PENGUJIAN INSERT

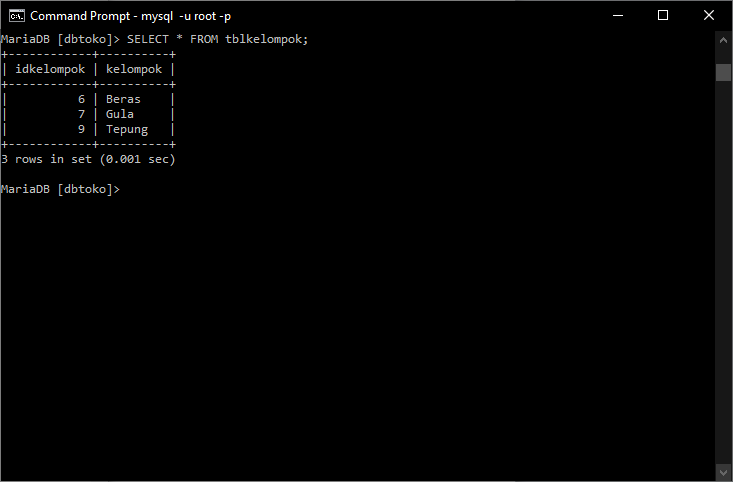
Insert pada tabel [tblbarang] HANYA BISA dilakukan menggunakan data yang tersedia pada tabel master. [Idkelompok] pada tabel master yang tersedia hanya [6,7,8]..

Pada pengujian INSERT idkelompok yang digunakan adalah 9 yang TIDAK TERSEDIA pada tabel master [tblkelompok].

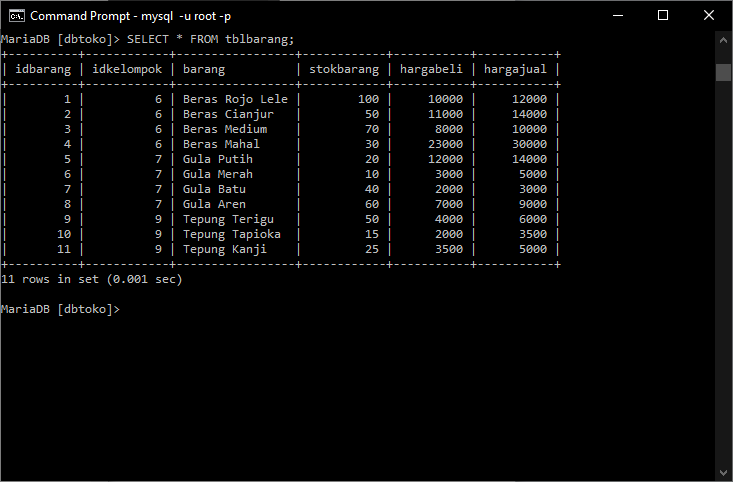


PENGUJIAN UPDATE

Coba lakukan update pada tabel master kemudian periksa di tabel transaksi. Ubah idkelompok tepung dari 8 menjadi 9 dengan perintah berikut;

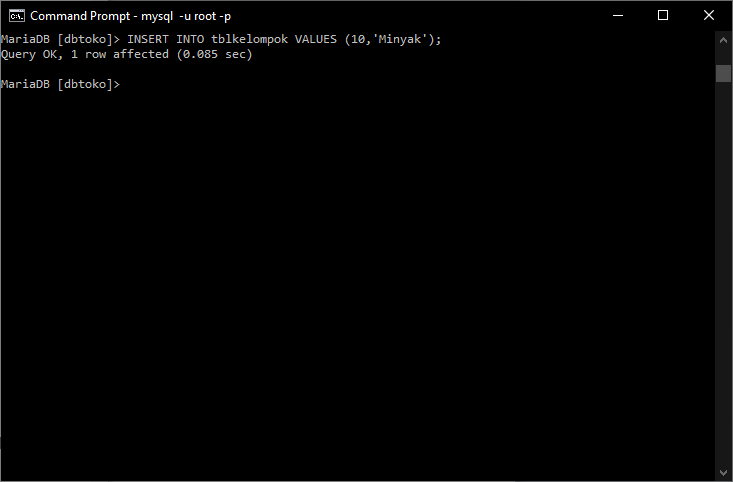


Periksa perubahan pada tblkelompok;



Periksa pada tblbarang;

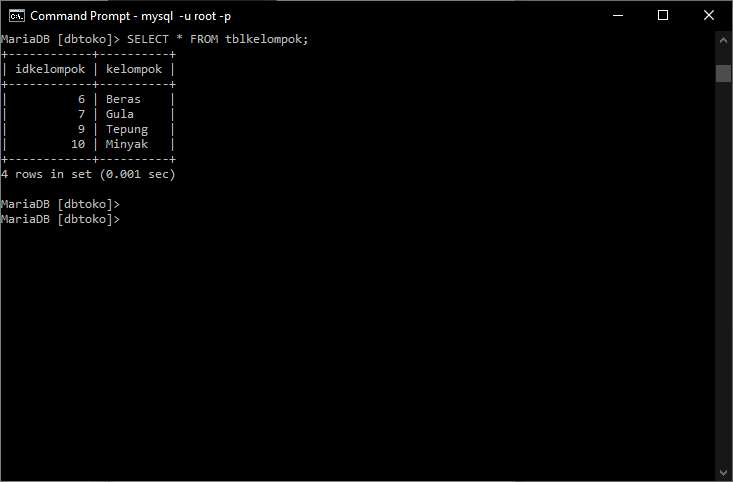
Pada ON UPDATE CASCADE jika kolom tabel master berubah maka kolom tabel transaksi juga berubah



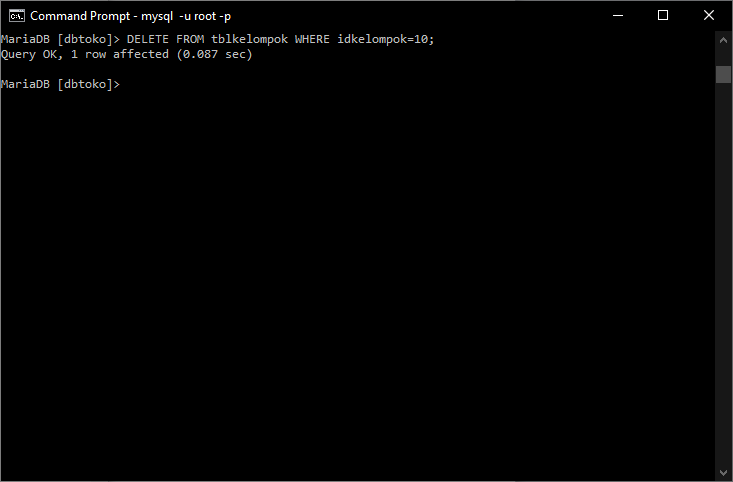
PENGUJIAN DELETE

Pengujian delete dilakukan dengan menghapus tabel master.

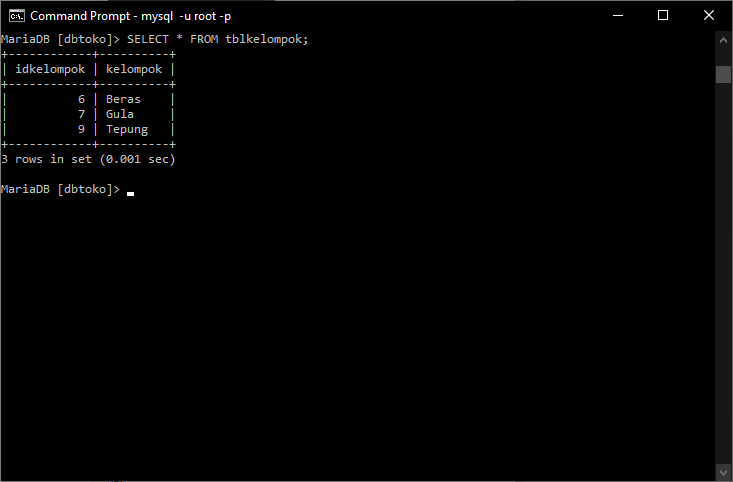
Karena [ idkelompok] 9 sudah digunakan pada tabel transaksi maka data tersebut tidak bisa dihapus seuai dengan relasi yang dibuat yaitu ON DELETE RESTRICT. Data yang bisa dihapus hanya yang belum digunakan pada tabel transaksi. coba INSERT kan data baru pada tabel master;



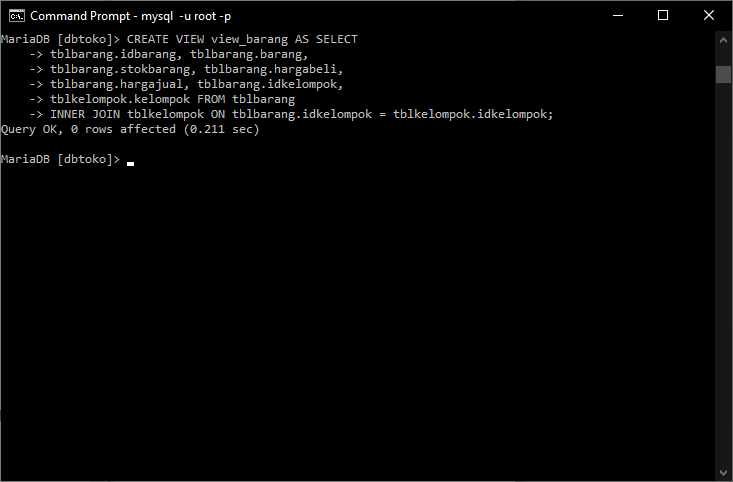
Periksa data yang sudah dimasukan



Karena kolom [idkelompok] 10 belum digunakan pada tabel transaksi maka kolom tersebut bisa dihapus. Lakukan penghapusan menggunakan perintah berikut;



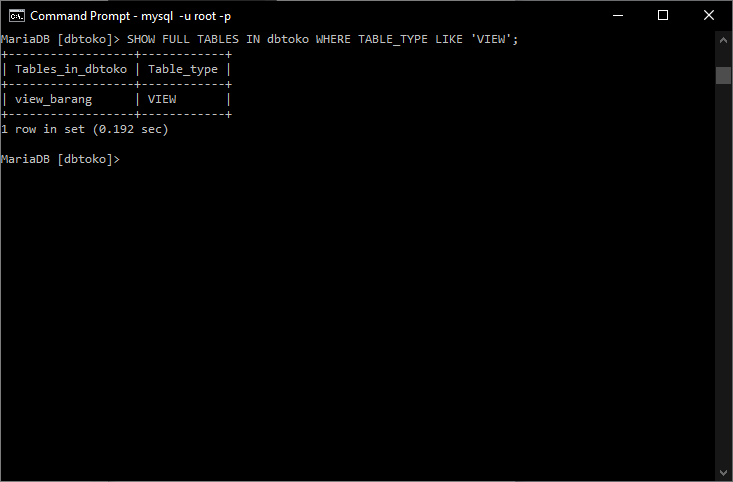
Periksa tabel yang sudah dihapus;



PEMBUATAN VIEW

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

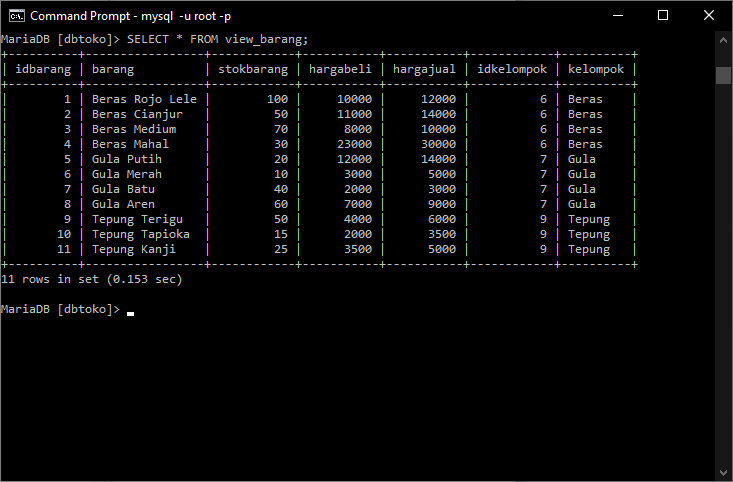
**CREATE VIEW nama\_view AS SELECT tabel\_transaksi.kolom\_transaksi, tabel\_master.kolom\_master FROM tabel\_transaksi INNER JOIN tabel\_master ON tabel\_transaksi.kolom\_transaksi = tabel\_master.kolom\_master;**



MENAMPILKAN SEMUA VIEW YANG SUDAH DIBUAT

View yang sudah dibuat bisa dilihat dengan perintah;

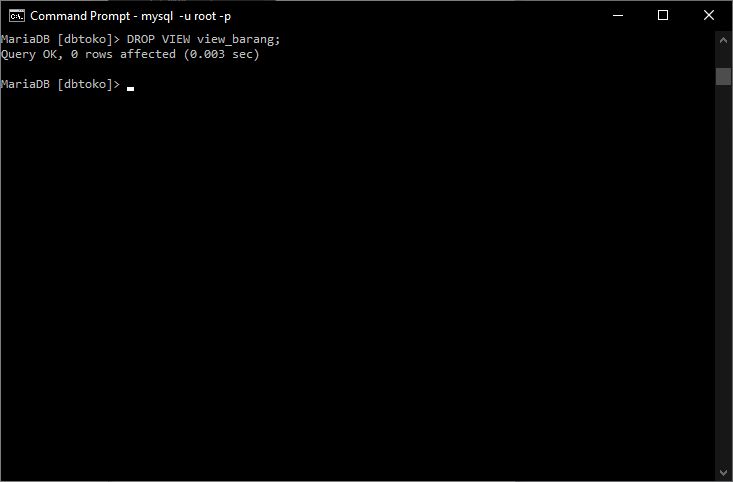
**SHOW FULL TABLES IN nama\_database WHERE TABLE\_TYPE LIKE ‘VIEW’;**



MELIHAT ISI VIEW

Untuk melihat isi VIEW sama dengan melihat isi tabel

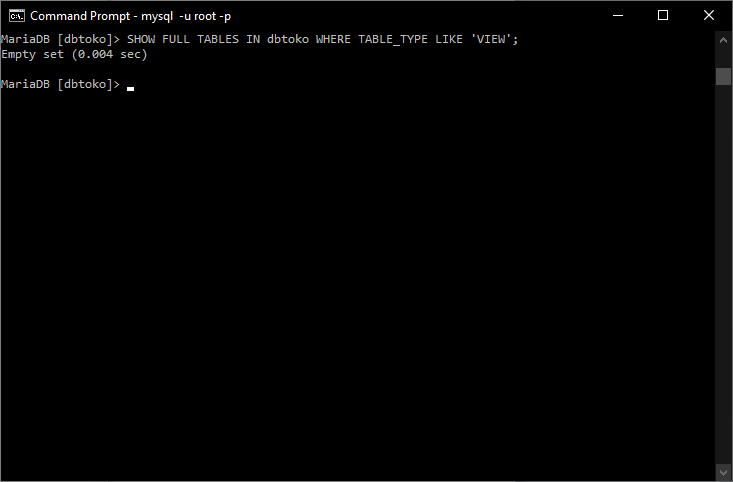
**SELECT \* FROM nama\_view;**



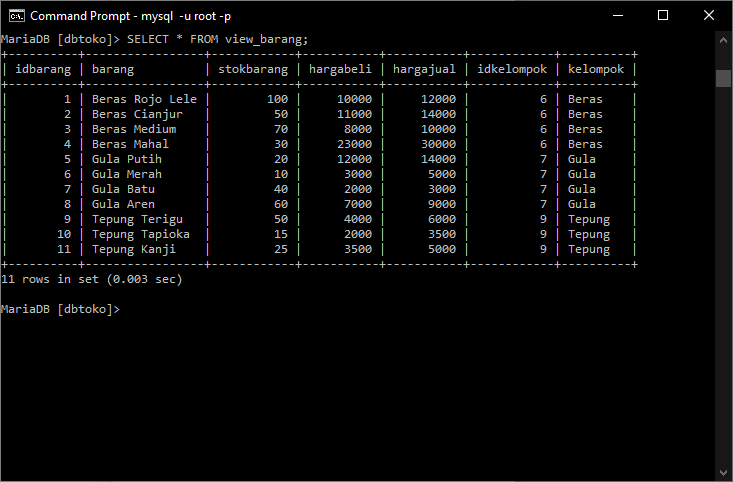
MENGHAPUS VIEW

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

**DROP VIEW nama\_view;**



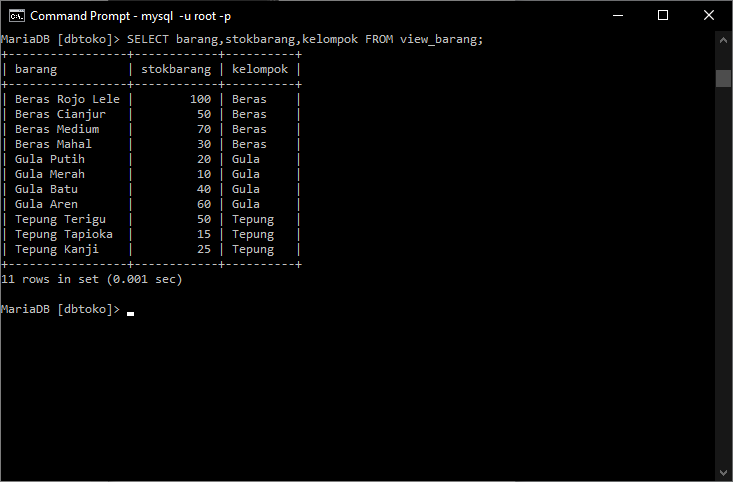
Periksa daftar view dengan perintah berikut;



SELECT SEMUA KOLOM (\*)

Select semua adalah menampilkan semua kolom dari tabel atau VIEW

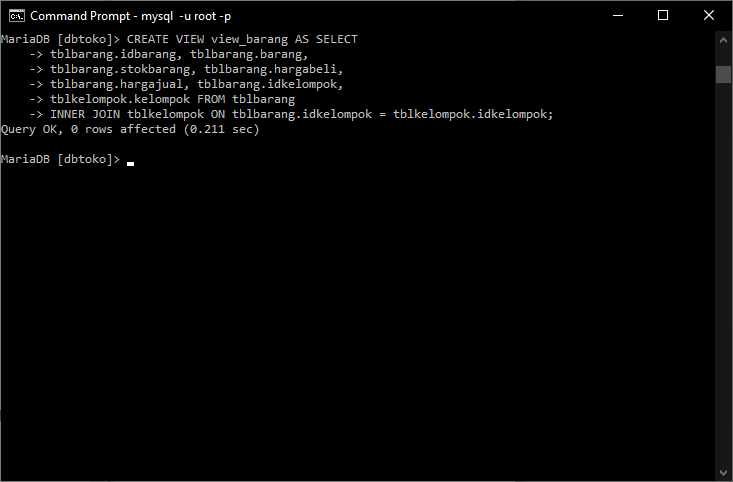
**SELECT \* FROM nama\_tabel\_atau\_nama\_view;**



SELECT SEBAGIAN KOLOM

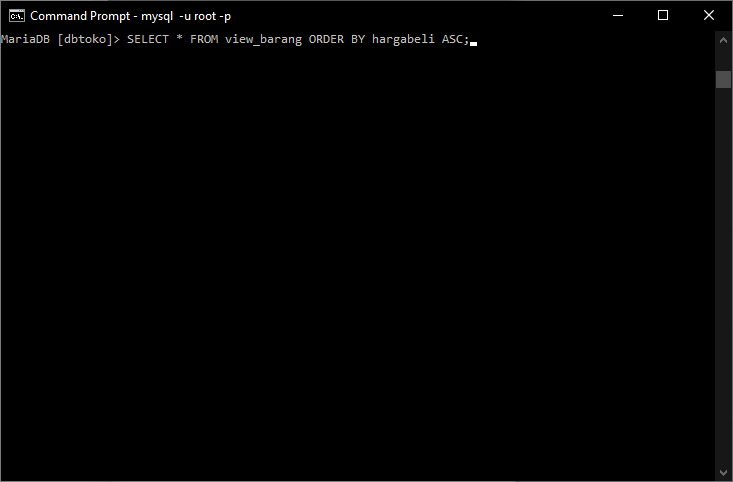
Select sebagian kolom adalah menampilkan hanya pada kolom yang dipilih

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



SELECT ORDER

Jika VIEW pada latihan diatas sudah dihapus, silahkan buat ulang untuk latihan materi selanjutnya;

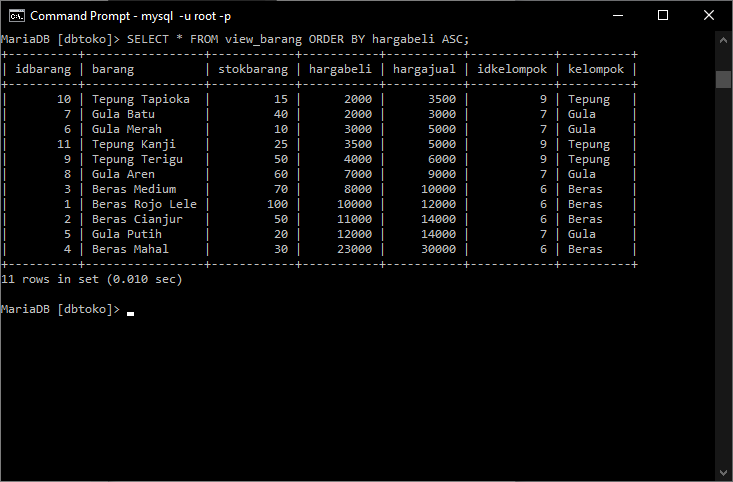


SELECT ORDER adalah cara untuk menampilkan data dalam urutan naik atau turun

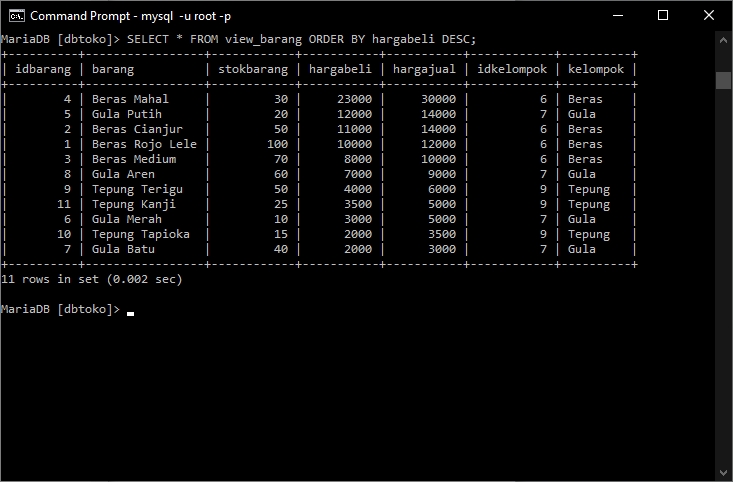
Jika NAIK menggunakan ASC

Jika TURUN menggunakan DESC

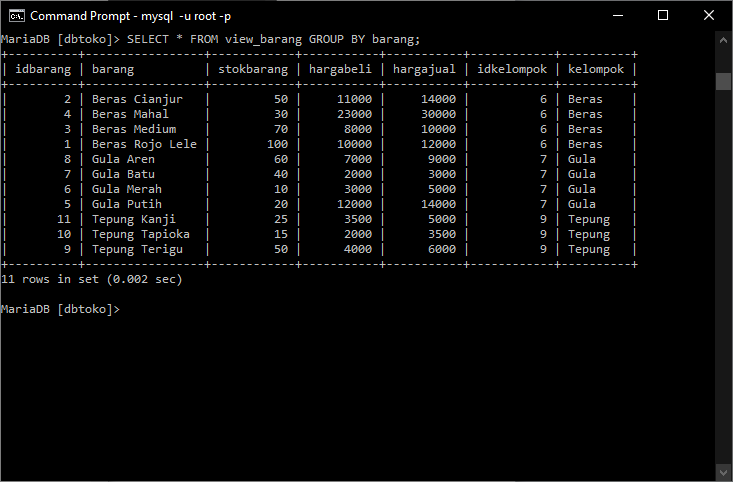
**SELECT \* FROM nama\_tabel\_atau\_view ORDER BY nama\_kolom ASC;**



Hasil select ORDER BY nama\_kolom ASC



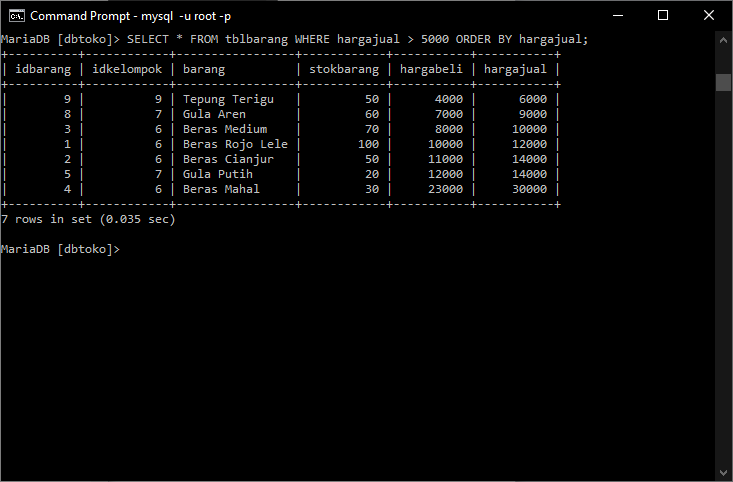
Hasil select ORDER BY nama\_kolom DESC

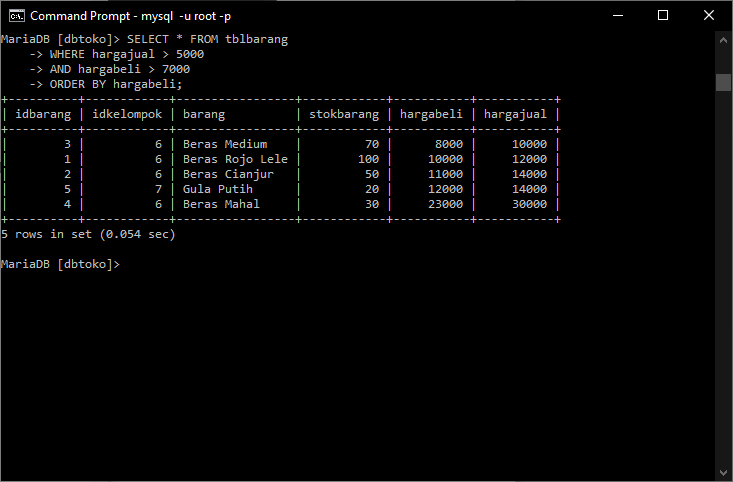


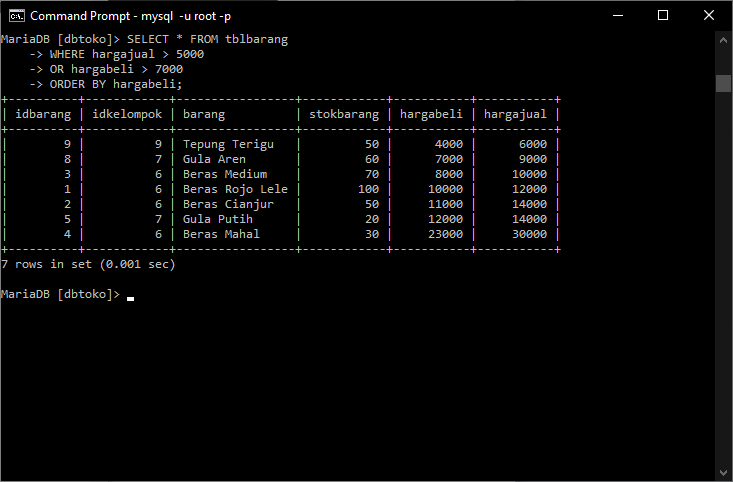
SELECT GROUP

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

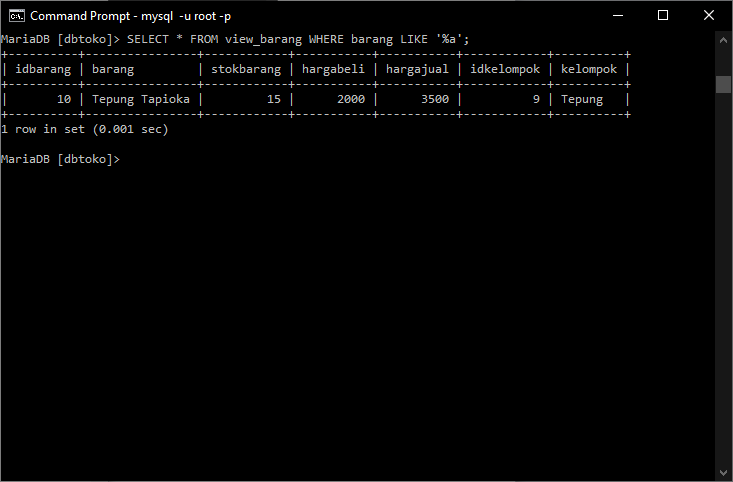
• Operator pembanding [=, >, <>, >=, <=, LIKE]

• Operator LOGIKA (AND, OR)

• SELECT

Contoh pengujian

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

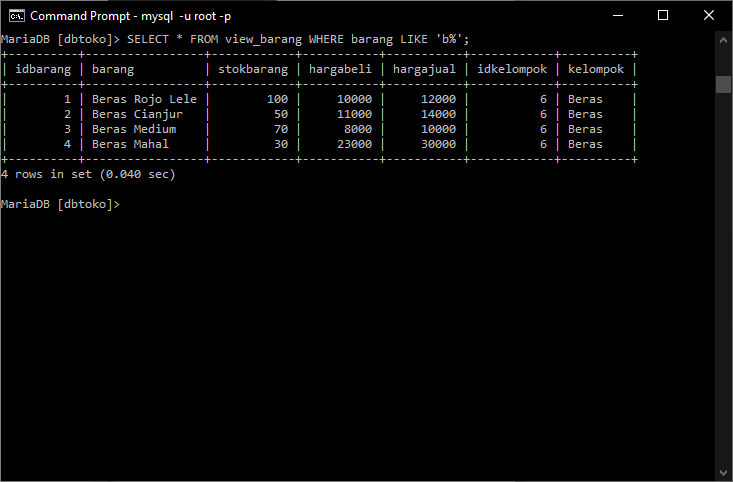


**SELECT \* FROM nama\_tabel\_atau\_view WHERE nama\_kolom LIKE ‘%\_apapun’**;

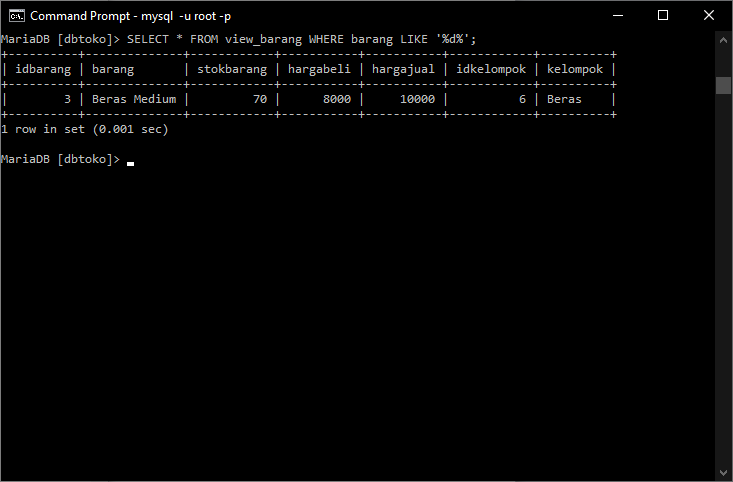
% adalah sebutan untuk sembarang atau apapun

Contoh:

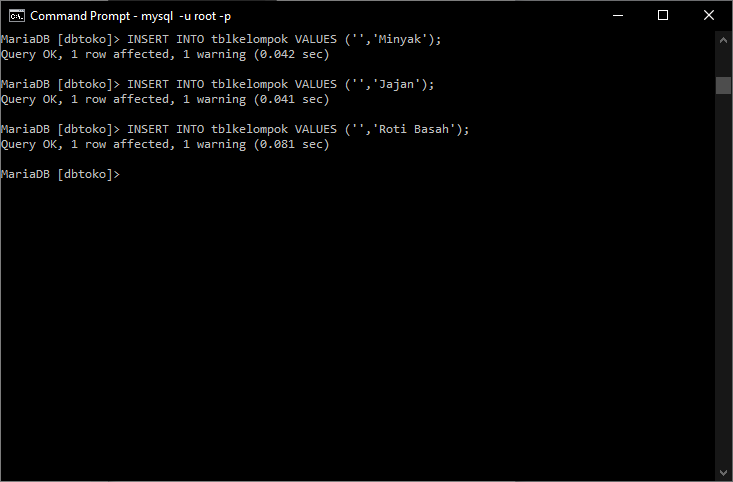
**%a artinya awalan sembarang yang penting akhiranya a**



**b% artinya awalan b akhiran sembarang**

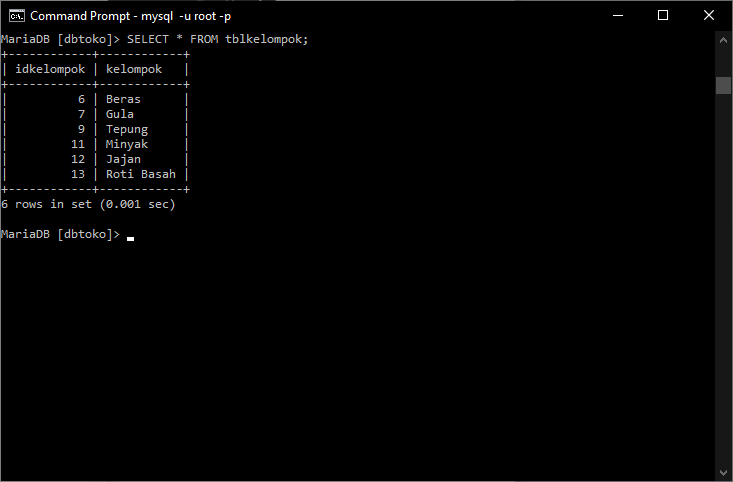


**%d% artinya awalan sembarang dan akhiran sembarang yang tengahnya ada huruf d**

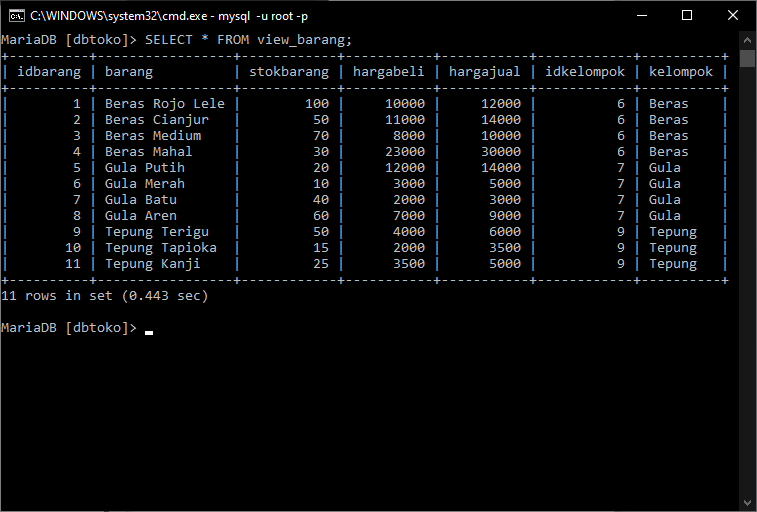


SUBQUERY (SELECT IN SELECT)

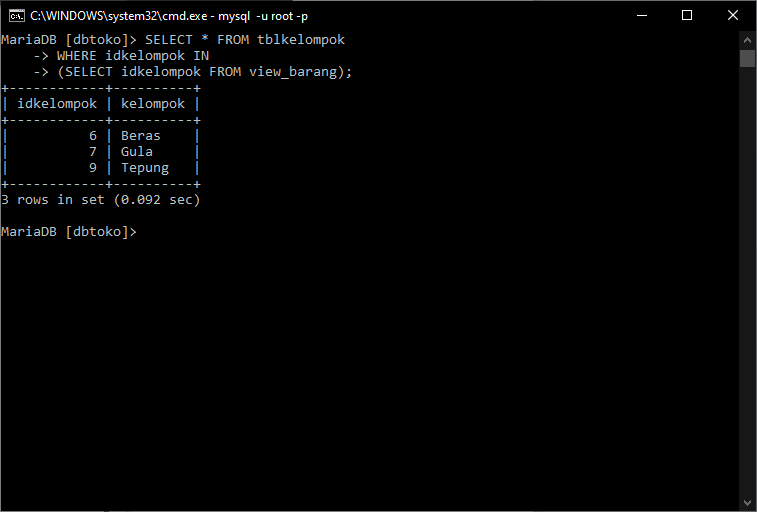
Subquery adalah SELECT yang ada di dalam SELECT. Sebelum berlatih menggunakan subquery tambahkan data pada [tblkelompok].



sehingga datanya akan tampil seperti berikut;



Periksa [view\_barang]



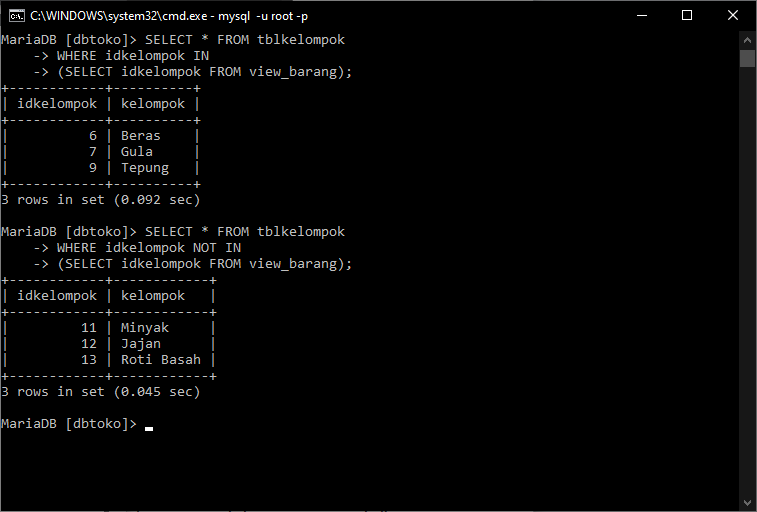
Pada [view\_barang] ada kelompok yang sudah menggunakan data dari tabel [tblkelompok]. Ada pertanyaan yang muncul dari data diatas adalah:

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

2. Ada berapa data yang BELUM digunakan pada VIEW?

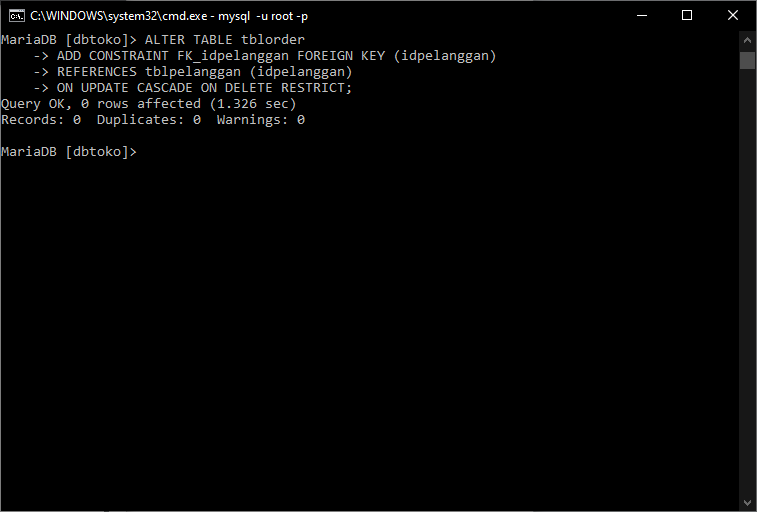
Untuk menjawab pertanyaan diatas bisa menggunakan SUBQUERY.

**SELECT \* FROM nama\_tabel WHERE nama\_kolom IN (SELECT nama\_kolom FROM nama\_tabel)**



**SELECT \* FROM nama\_tabel WHERE nama\_kolom NOT IN (SELECT nama\_kolom FROM nama\_tabel)**

**PENGGUNAAN TANDA \* PADA SELECT IN HANYA BISA DILAKUKAN PADA SELECT YANG PERTAMA**

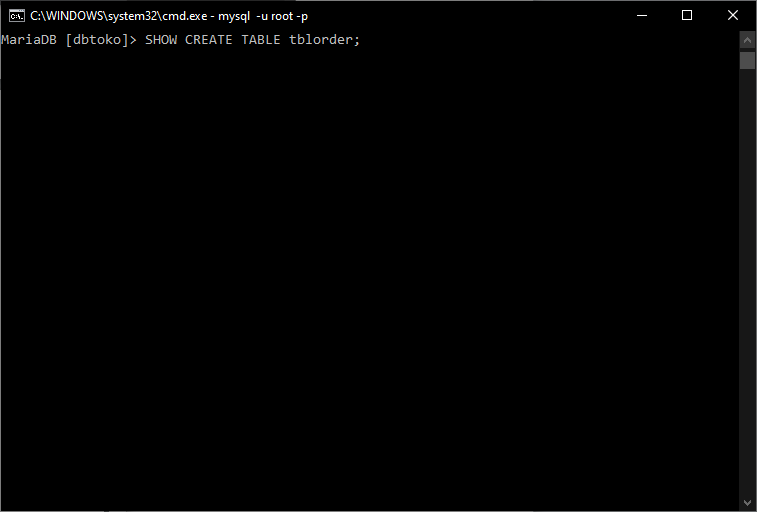


MEMBUAT RELASI ANTAR TABEL

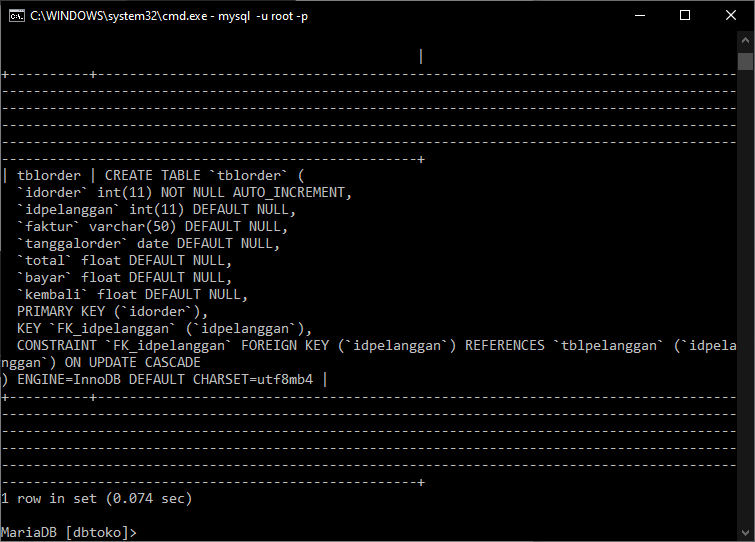
pada desain database terdapat relasi antar tabel [tblpelanggan] dan tabel [tblorder]. Lengkapi proses pembuatan relasi pada tabel [tblorder].

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 dengan perintah berikut;



Hasil pembuatan relasi



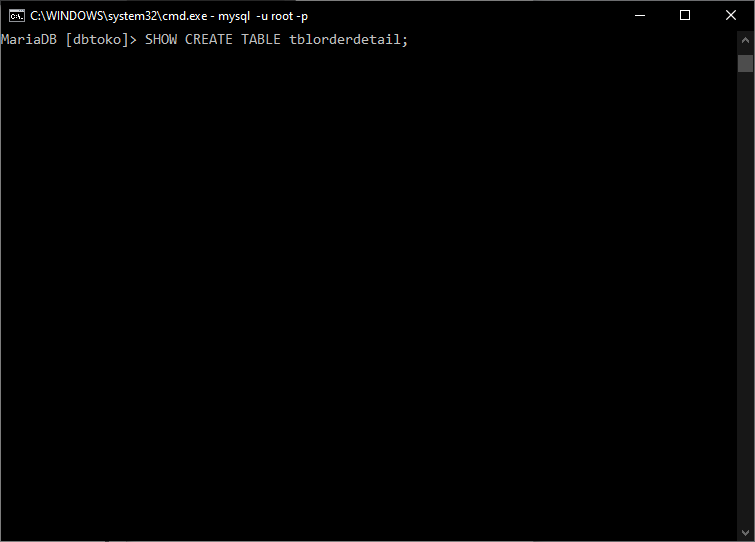
RELASI 2 TABEL MASTER DAN 1 TABEL TRANSAKSI

Pada tabel [tblorderdetail] detail terdapat relasi antara tabel [tblorder] dan tabel [tblbarang]

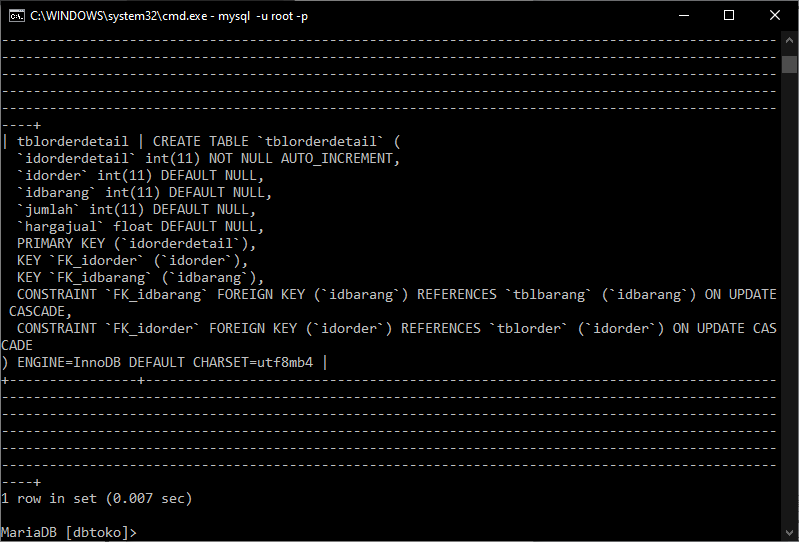
Buat relasi antara tabel [tblorderdetail] dan tabel [tblorder]



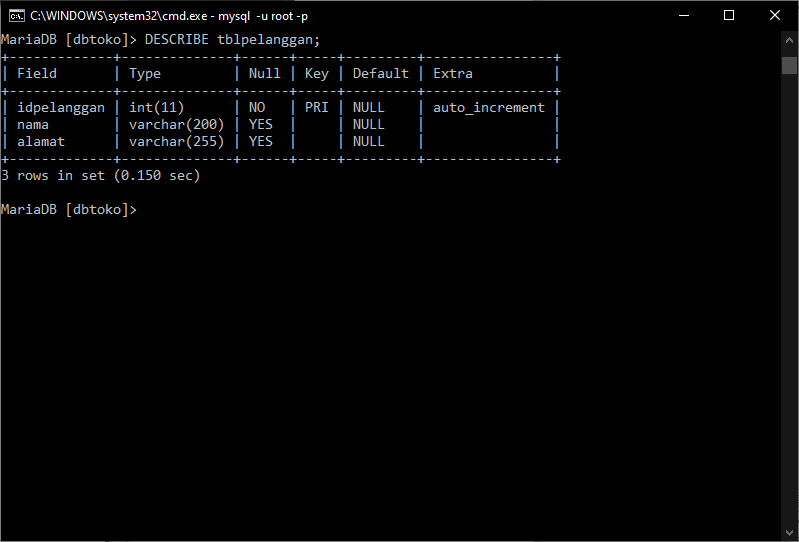
Buat relasi antara tabel [tblorderdetail] dan tabel [tblbarang]



Periksa hasil pembuatan relasi antara 2 tabel master [tblorder] [tblbarang] dan 1 tabel transaksi [tblorderdetail]



Hasil pembuatan relasi



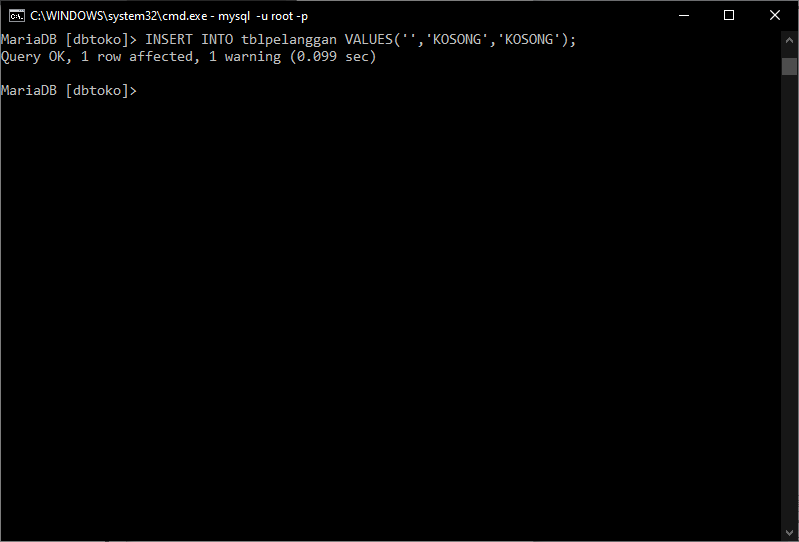
DUMMY DATA

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

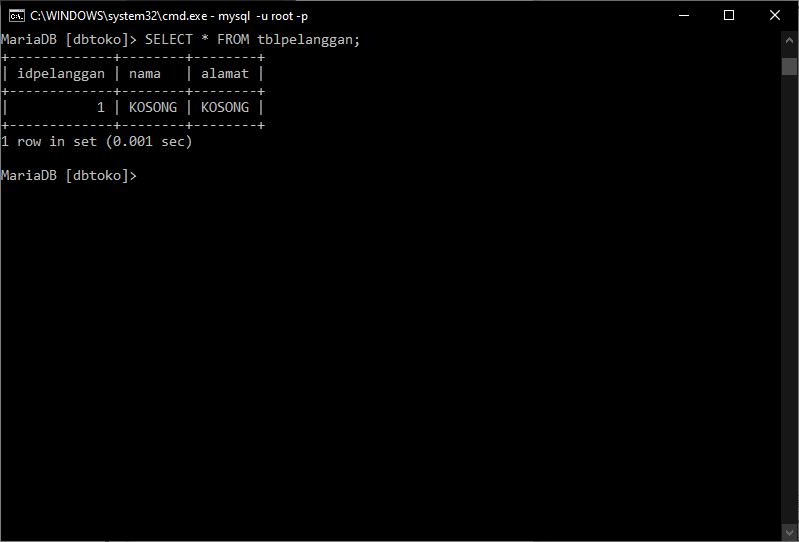
Pada materi ini akan menggunakan dummy data pada nama pelanggan. Umumnya pelanggan mempunyai nama. Jika anda berbelanja pada supermarket atau minimarket maka anda tidak akan ditanya tentang nama. Anda hanya akan ditanya yang anda beli. Pertanyaannya bagaimana aplikasi mencatat nama anda?

Pada tabel [tblpelanggan] aplikasi akan di isi dengan dummy data dengan nama pelanggan sebegai berikut;

Tampilkan struktur tabel terlebih dahulu

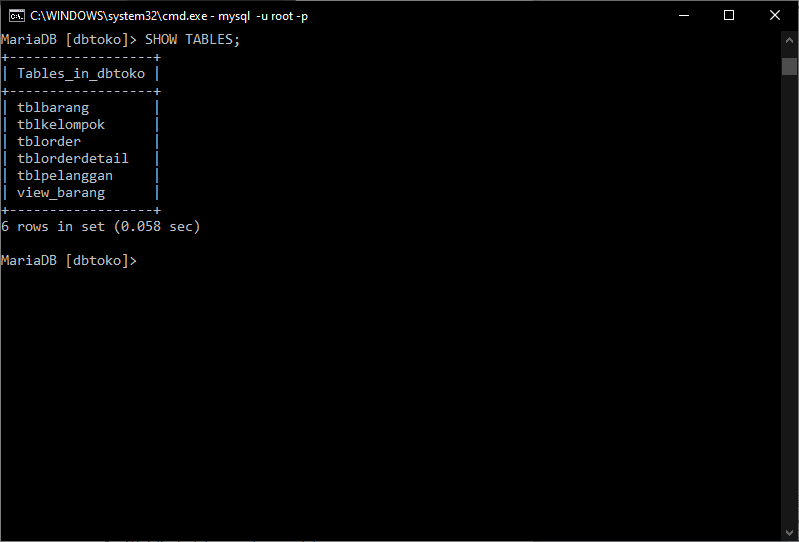


Isikan nama pelanggan dengan dummy data



Periksa hasil INSERT datanya;

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

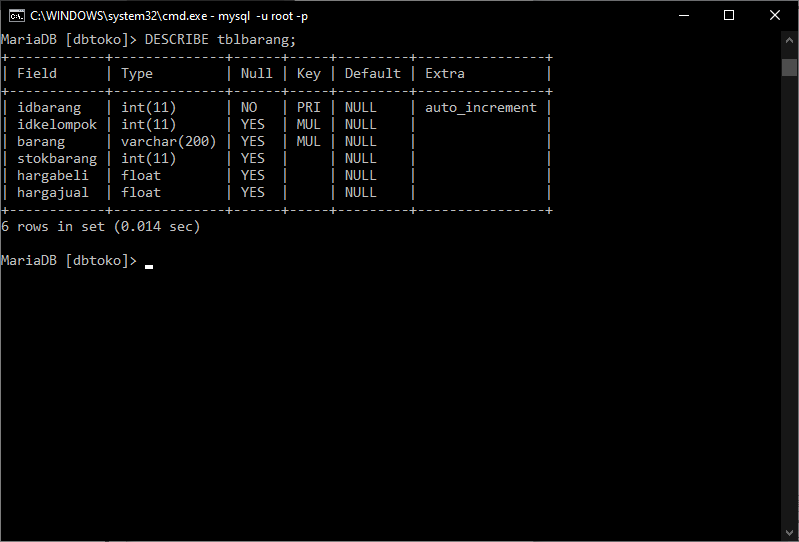
• BEFORE DELETE (Sebelum DELETE) pada tabel yang dimaksud

• AFTER UPDATE (Setelah UPDATE) pada tabel yang dimaksud

• BEFORE UPDATE (Sebelum UPDATE) pada tabel yang dimaksud

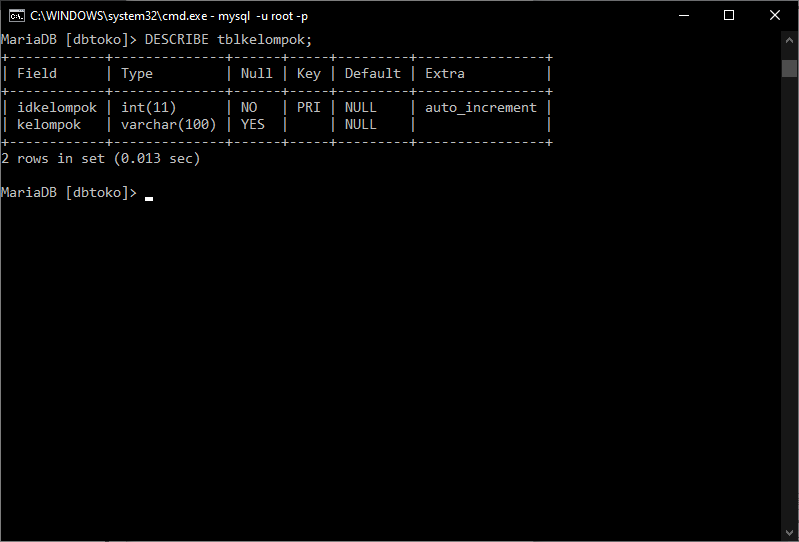
Sebelum membuat trigger pastikan anda memahami proses bisnis dari pembuatan database yang anda lakukan. Sebagai latihan buka semua database yang sudah dibuat pada materi sebelumnya, ikuti langkah berikut;

1. TAMPILKAN SEMUA TABEL

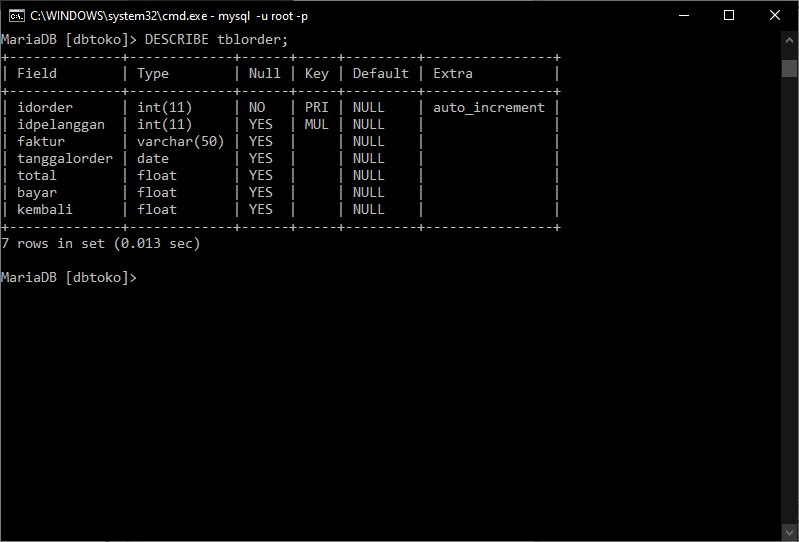


2. LIHAT STRUKTUR SEMUA TABEL

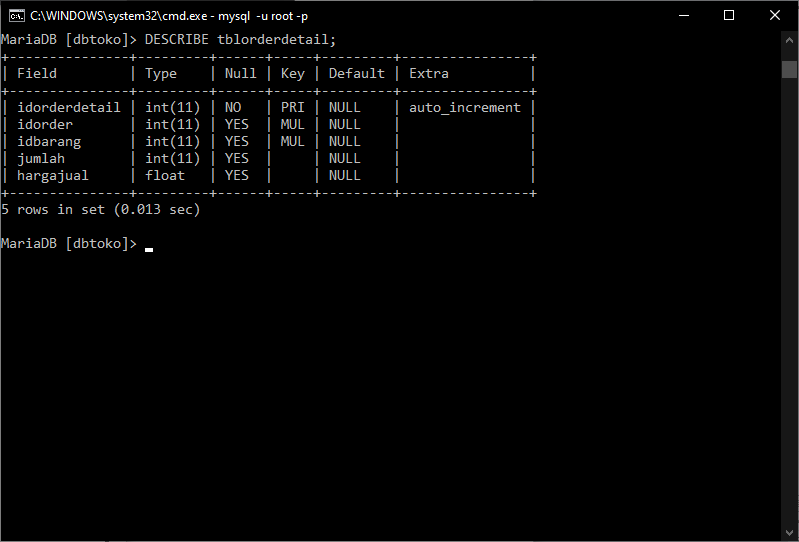
[tblbarang]



[tblkelompok]



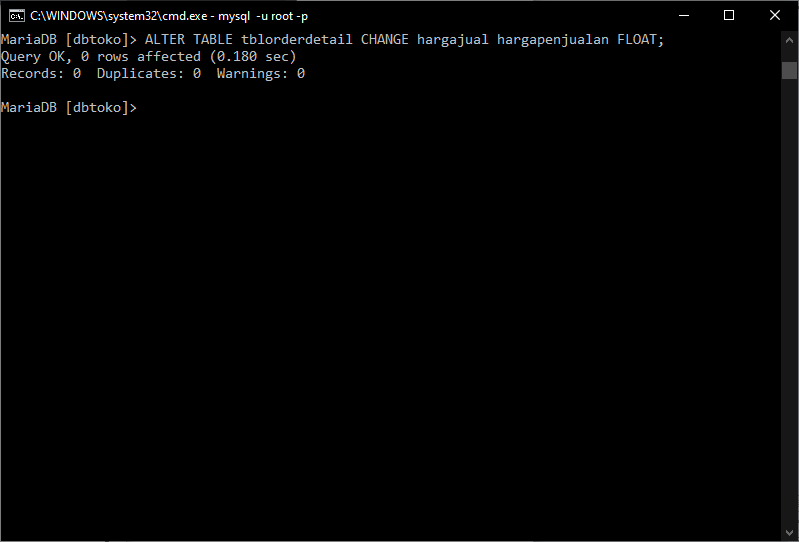
[tblorder]



[tblorderdetail]



[tblpelanggan]



3. PENYUSUNAN PROSES BISNIS

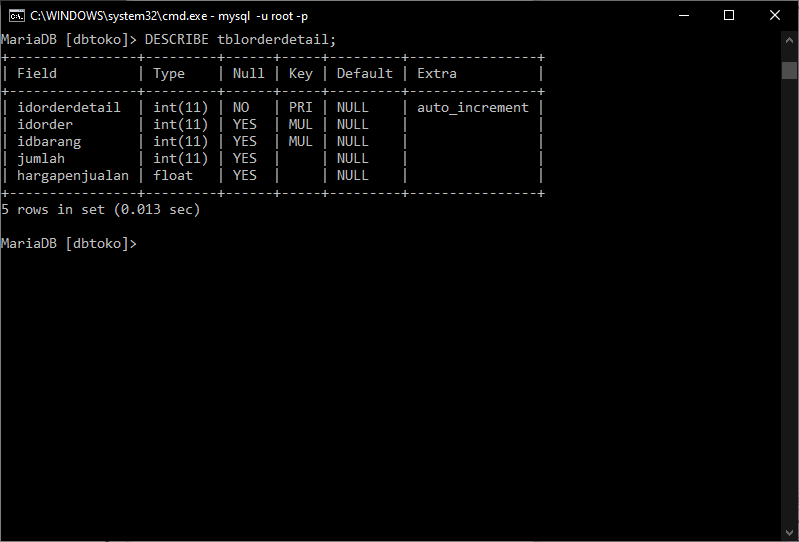
kolom [stokbarang] pada tabel [tblbarang] akan BERKURANG jika terjadi INSERT data pada tabel [tblorderdetail]

kolom [stokbarang] pada tabel [tblbarang] akan BERTAMBAH jika terjadi DELETE data pada tabel [tblorderdetail]

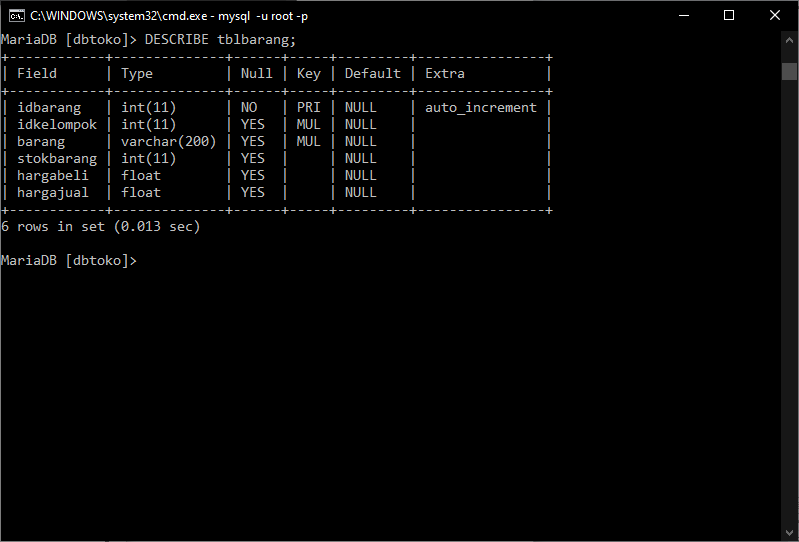
kolom [total] pada tabel [tblorder] akan BERTAMBAH jika terjadi INSERT data pada tabel [tblorderdetail] kolom [total] pada tabel [tblorder] akan BERKURANG jika terjadi DELETE data pada tabel [tblorderdetail]

4. PEMERIKSAAN NAMA KOLOM YANG SAMA

Sebelum membuat TRIGGER pastikan tidak ada nama kolom yang sama pada setiap tabel. Jika melihat tabel diatas terdapat nama kolom yang sama. Yaitu [hargajual] pada tabel [tblbarang] dan [hargajual] pada tabel [tblorderdetail]. Ubah nama kolom [hargajual] di tabel [tblorderdetail] menjadi [hargapenjualan]



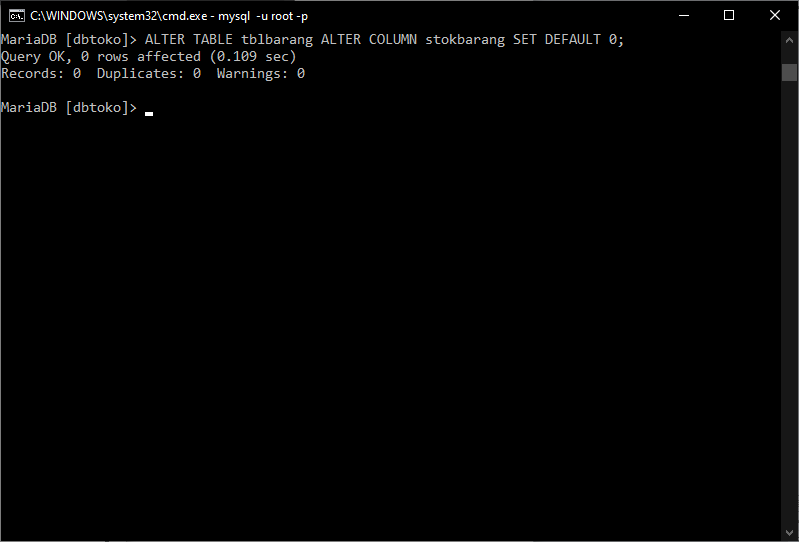
Lihat hasil perubahan nama kolom



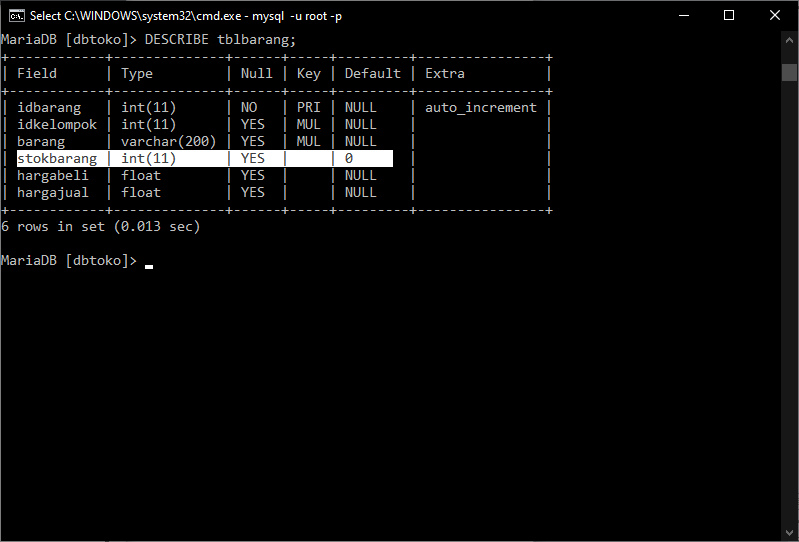
5. PEMERIKSAAN KOLOM UNTUK PERUBAHAN HASIL TRIGGER

Lakukan pemeriksaan pada pada kolom yang akan terkena dampak dari proses trigger. Jika yang diproses oleh trigger adalah operasi matematika (penjumlahan, pengurangan, pembagian, perkalian). Pastikan SET DEFAULT VALUE pada kolom tabel tersebut di isi dengan ANGKA NOL. Jika belum di isi angka NOL lakukan ALTER untuk merubah kolom tersebut. Lihat contoh berikut;

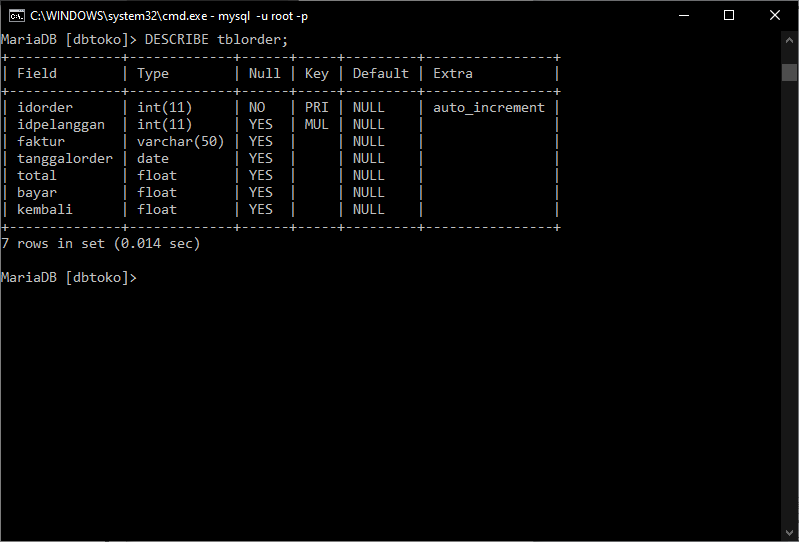
Kolom [stokbarang] belum di SET DEFAULT dengan ANGKA NOL



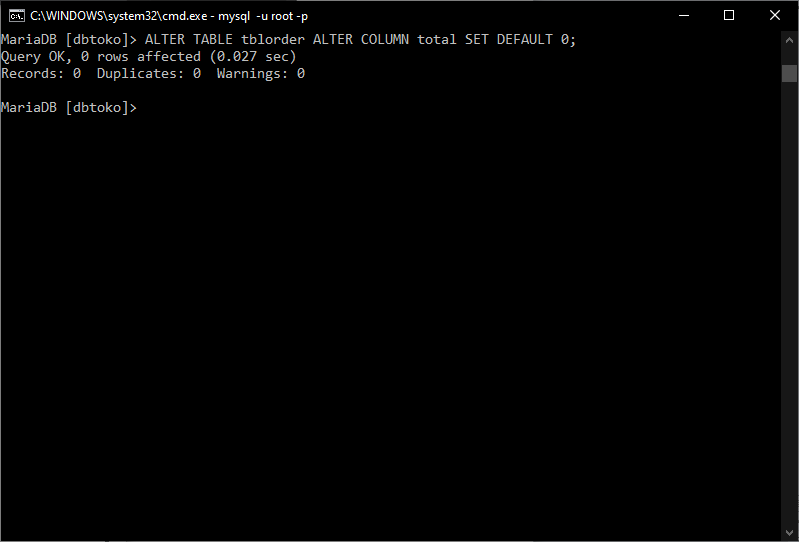
Perintah ALTER pada tabel [tblbarang]



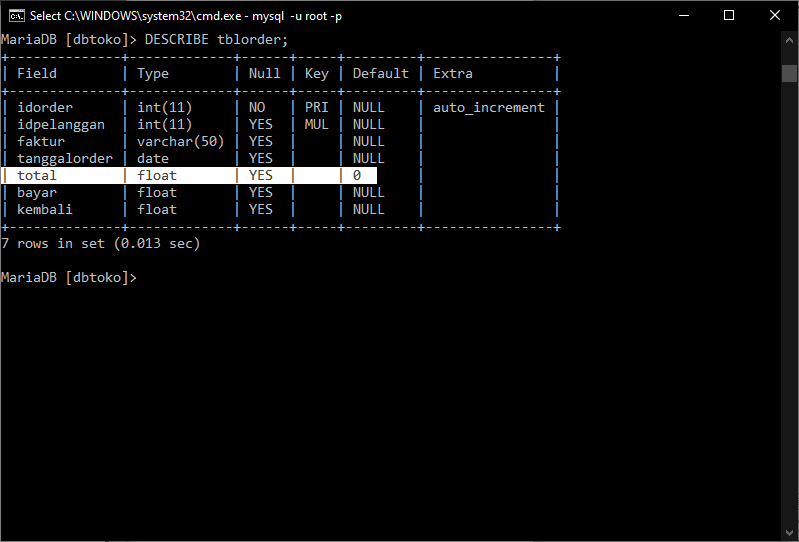
Hasil perubahan



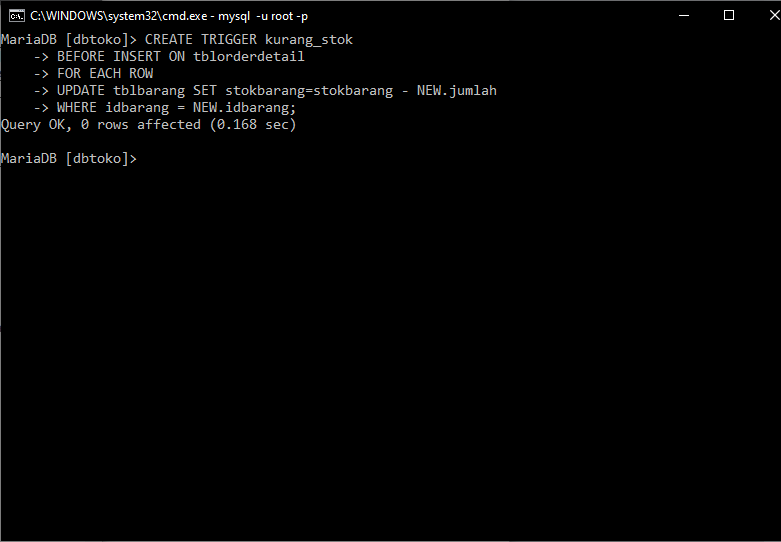
Pemeriksaan kolom [total] pada tabel [tblorder]. Belum di set DEFAULT ANGKA NOL



Lakukan perubahan dengan ALTER. Perintah ALTER pada tabel [tblorder]

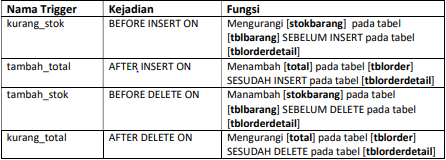


Hasil perubahan kolom [total] pada [tblorder]



PEMBUATAN TRIGGER

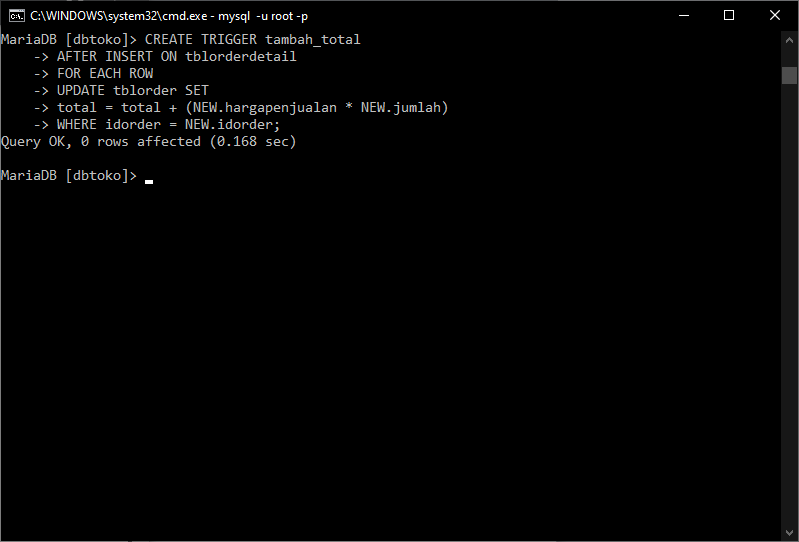
Setelah semua konsep, proses bisnis, dan pemeriksaan tabel dilakukan sekarang waktunya membuat TRIGGER. Ada 4 trigger yang akan dibuat.



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

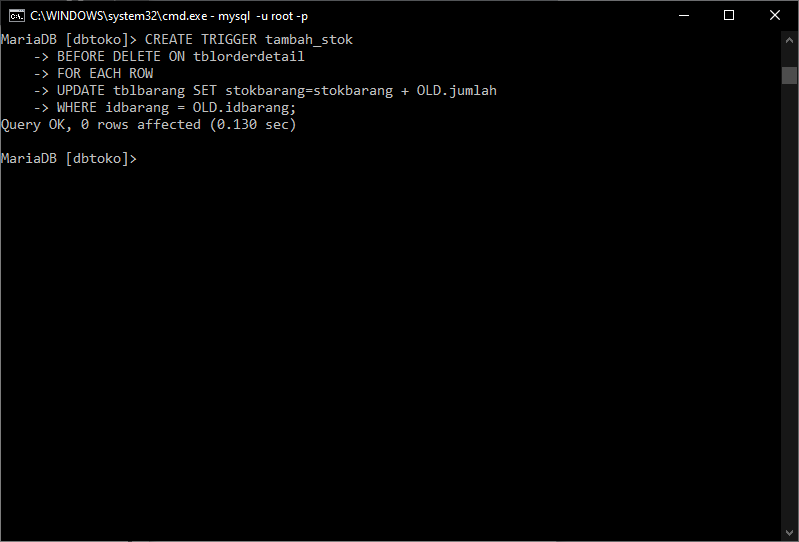
kurang\_stok

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 tabel [tblorderdetail] Contoh : jika stok barang 100 kemudian terjadi penjualan pada tabel [tblorderdetail] sebanyak 35 maka nilainya menjadi [stokbarang = 100 – 35] nilai selanjutnya [stokbarang] menjadi 65



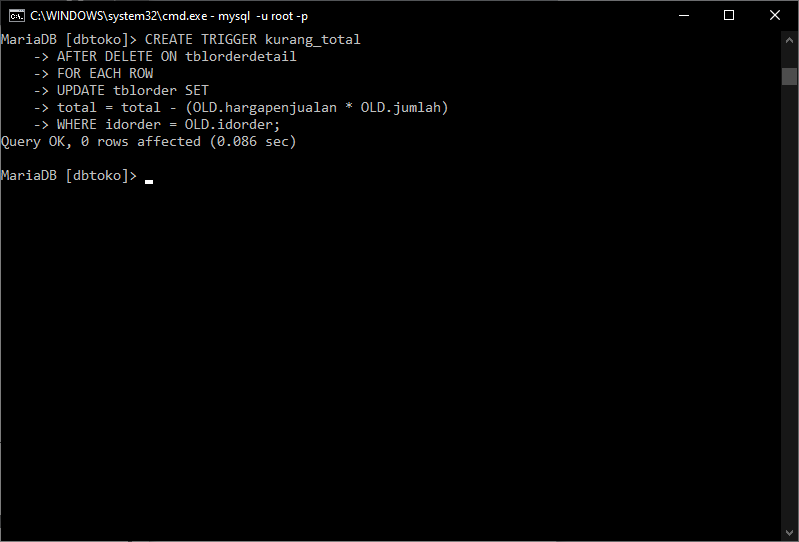
tambah\_total

PENJELASAN: Trigger [tambah\_total] digunakan untuk menambah nilai [total] pada tabel [tblorder] saat AFTER INSERT (sesudah insert) pada tabel [tblorderdetail]. [NEW.hargapenjualan \* NEW.jumlah] adalah data jumlah yang akan di jual dan harga penjualan yang AKAN MASUK pada kolom [jumlah] dan kolom [hargapenjualan] di tabel [tblorderdetail] Contoh : terjadi penjualan pada barang dengan [iDCLrang=1] sebanyak [jumlah = 35] dengan [hargapenjualan = 12000] maka [total=420000]



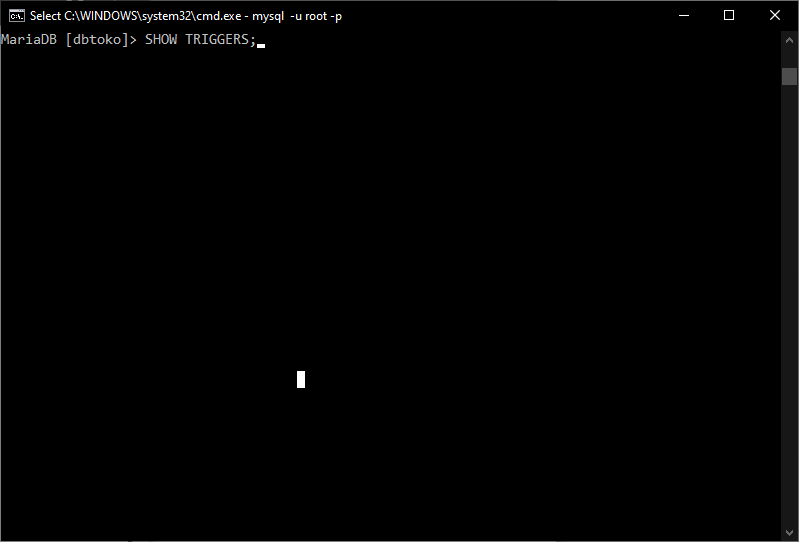
tambah\_stok

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] Contoh : jika stok barang 65 kemudian terjadi DELETE pada tabel [tblorderdetail] maka nilai yang [jumlah = 35] yang ada pada tabel [tblorderdetail] akan dijumlahkan dengan nilai [stokbarang] sehingga nilainya menjadi [stokbarang = 65 + 35] nilai selanjutnya [stokbarang] menjadi 100



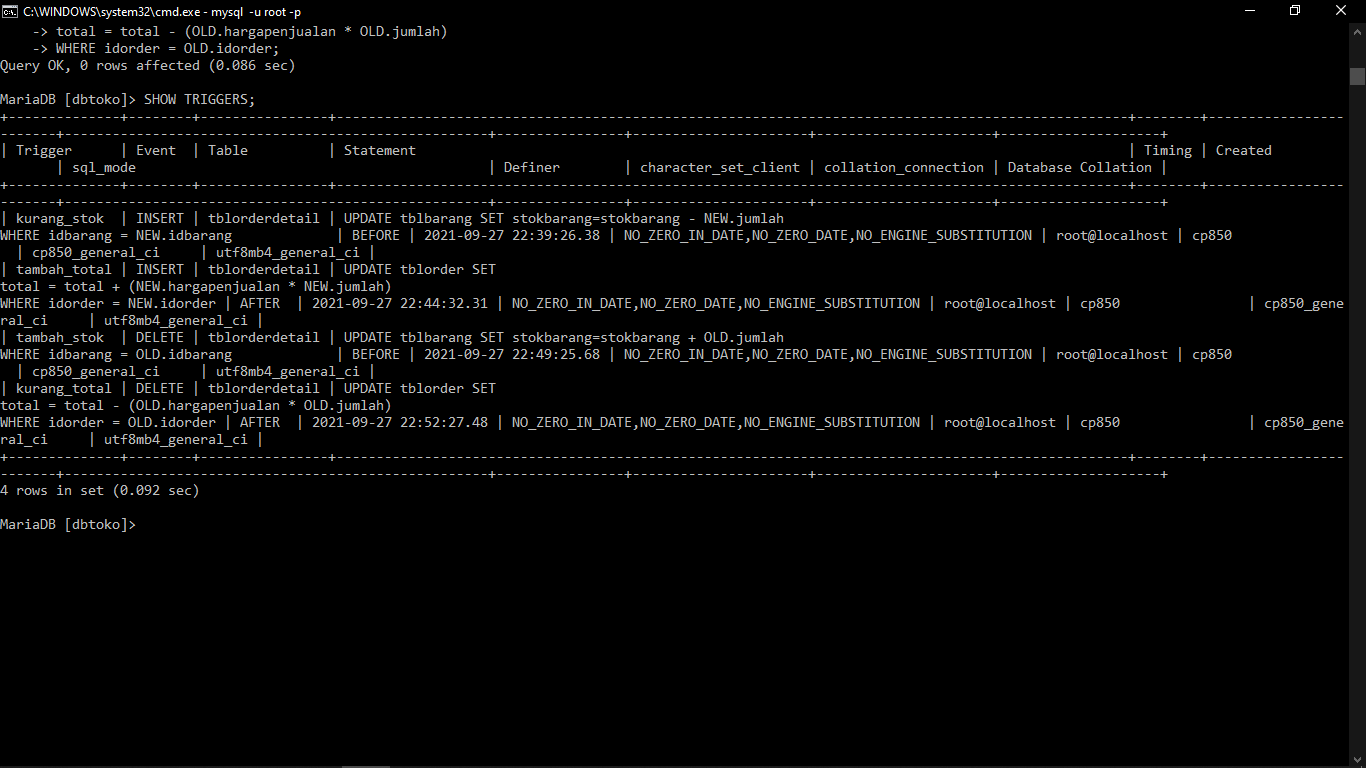
kurang\_total

PENJELASAN: Trigger [kurang\_total] digunakan untuk mengurangi nilai [total] pada tabel [tblorder] saat AFTER DELETE (sesudah DELETE) pada tabel [tblorderdetail]. [OLD.hargapenjualan \* OLD.jumlah] adalah data jumlah yang dan harga penjualan yang sudah ada pada tabel [tblorderdetail]

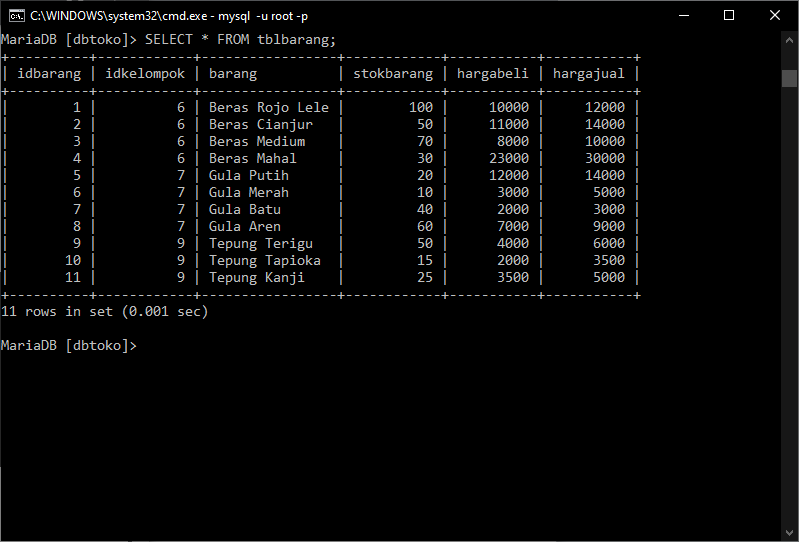


MENAMPILKAN TRIGGER

Trigger yang sudah dibuat bisa ditempilkan dengan cara



Tampilan hasil trigger yang dibuat;

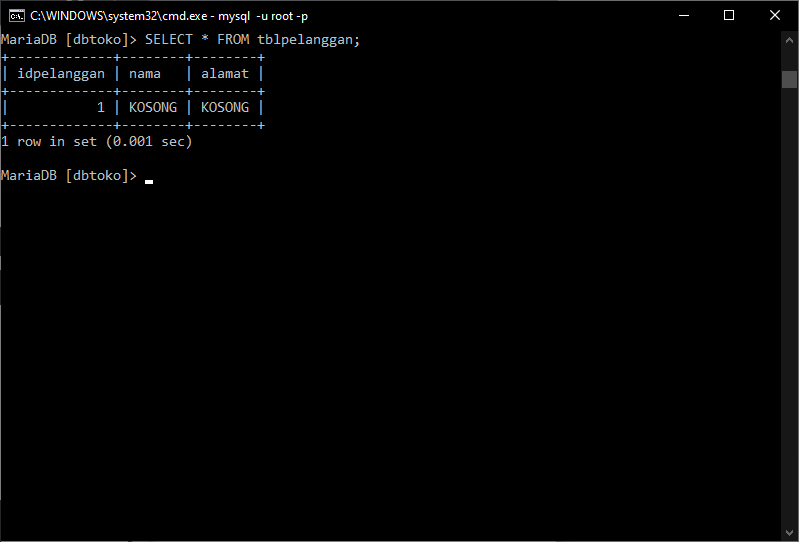


PENGUJIAN TRIGGER

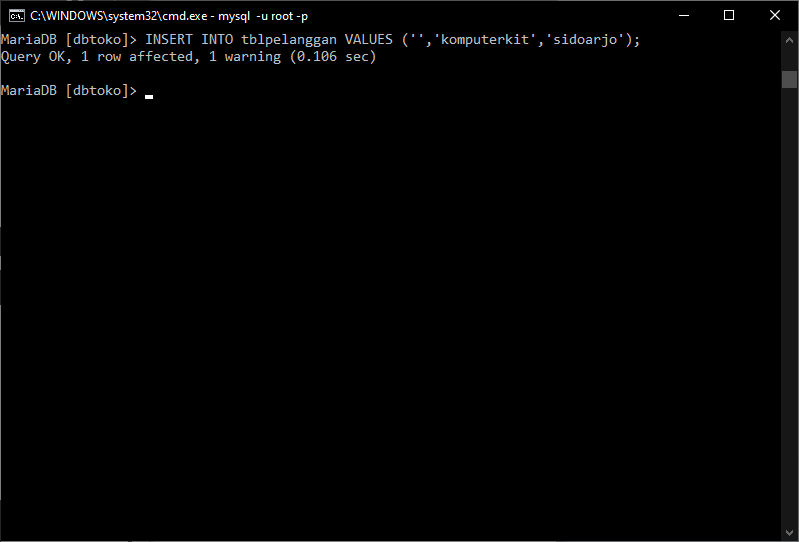
Setelah trigger dibuat maka langkah selanjutnya adalah menguji trigger;

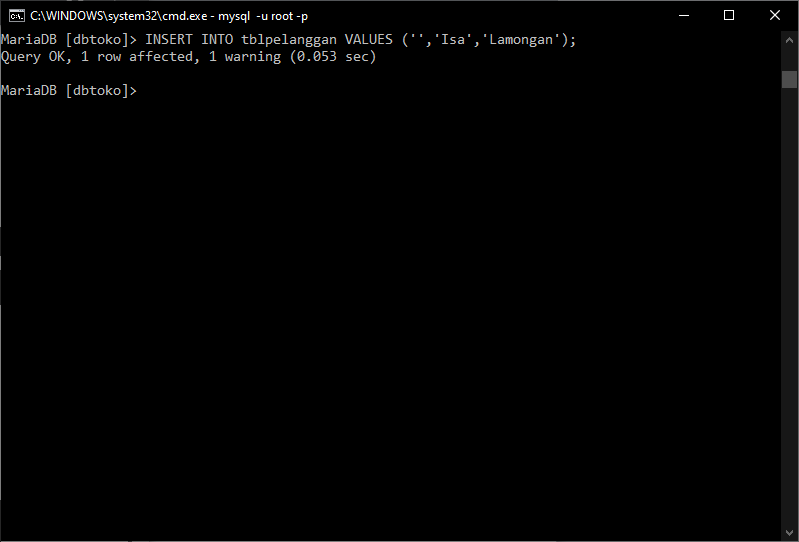
1. PERSIAPAN

Tampilkan tabel [tblbarang]

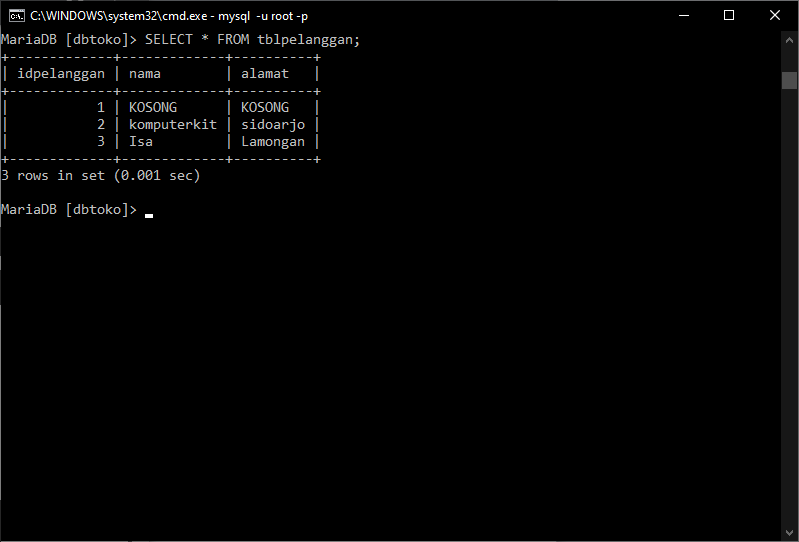


Tampilkan tabel [tblpelanggan]

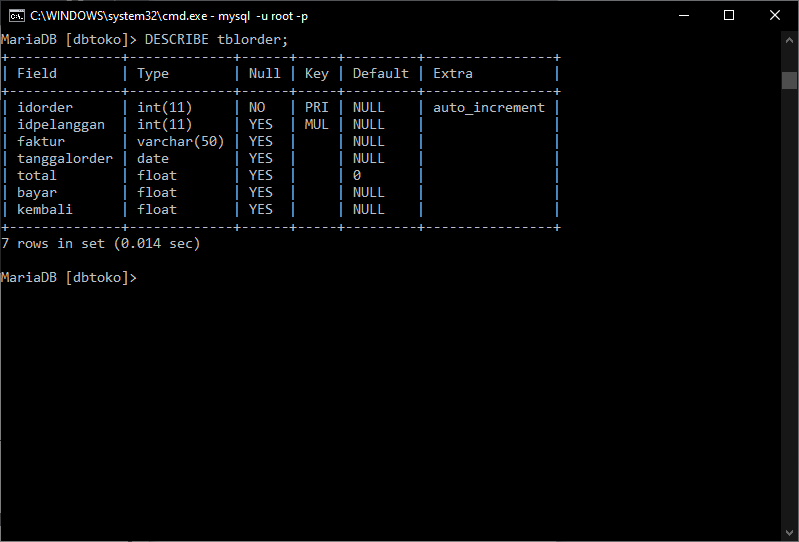




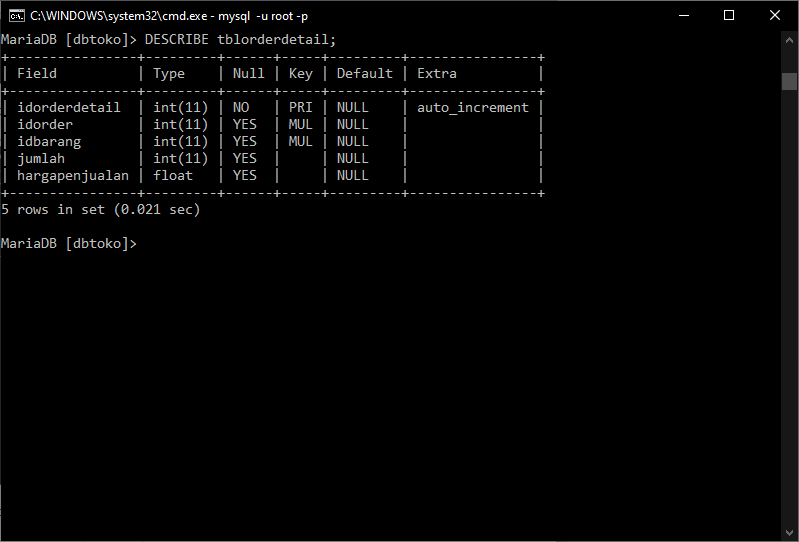
Karena data pelanggan hanya satu, tambahkan 2 pelanggan lagi untuk pengujian



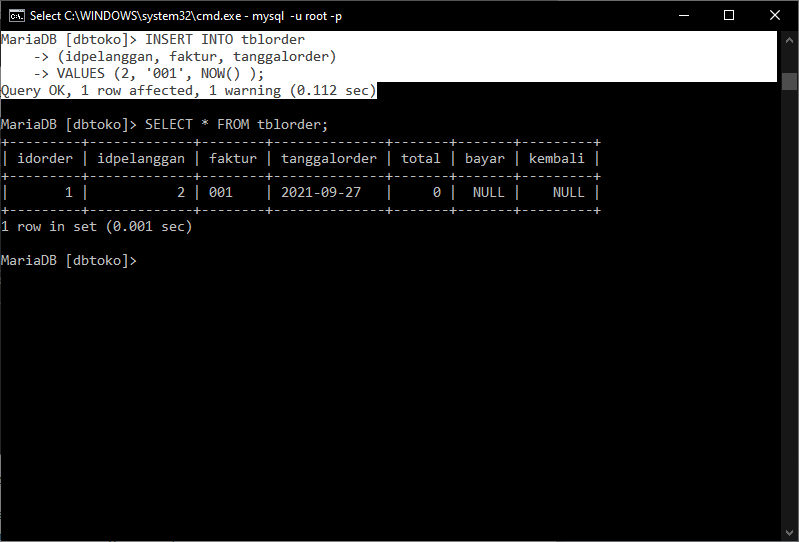
Tampilkan kembali data pelanggan



Tampilkan struktur tabel [tblorder]

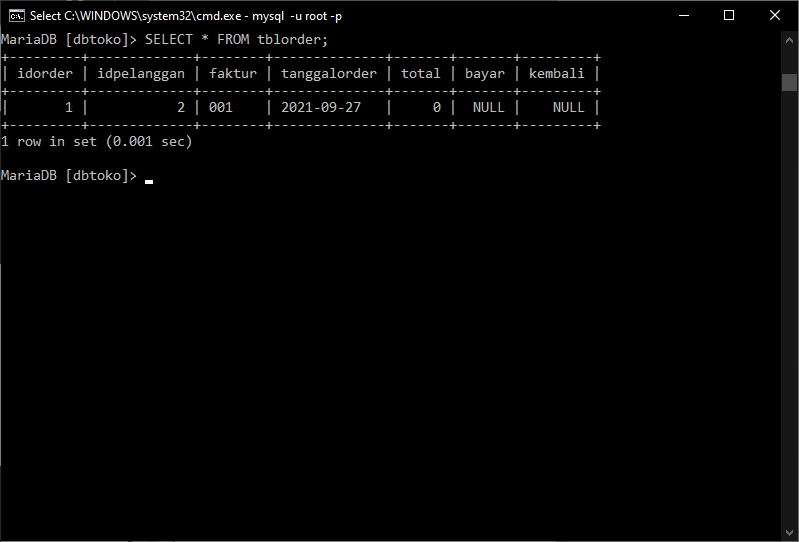


Tampilkan struktur tabel [tblorderdetail]

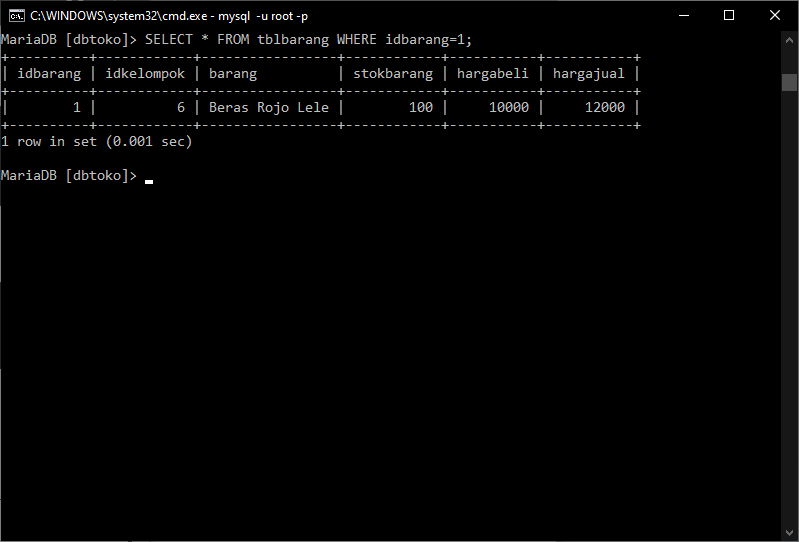


2. PENGUJIAN INSERT PADA TABEL

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

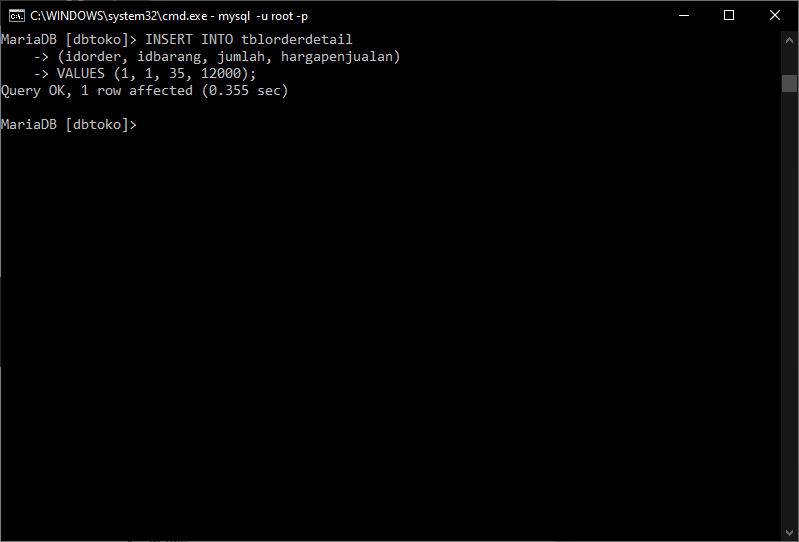


Periksa dengan perintah sebagai berikut;

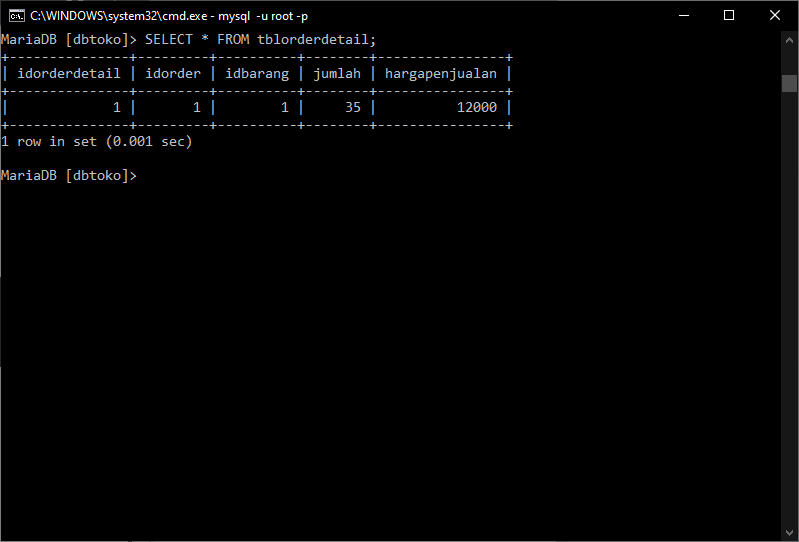


Buat INSERT data pada tabel [tblorderdetail] dengan menggunakan [idorder = 1] sesuai yang ada di tabel [tblorder] dengan [iDCLrang = 1] pada tabel [tblbarang].

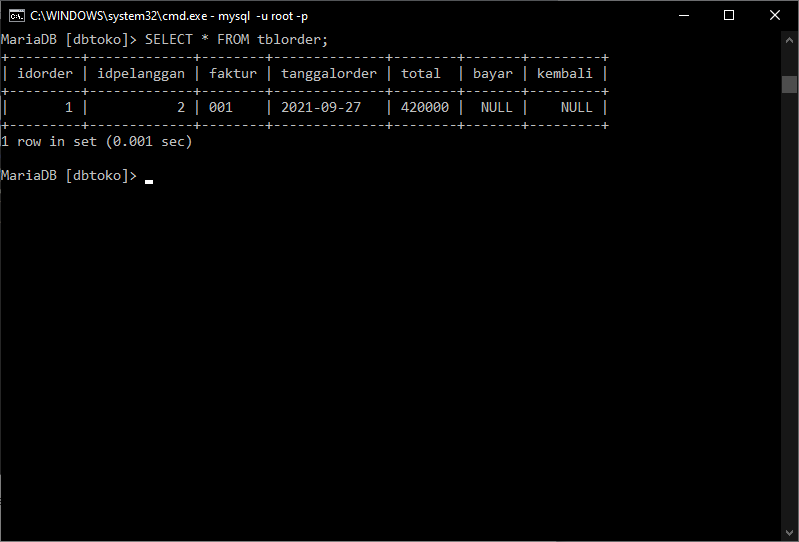
Tabel SEBELUM PROSES INSERT pada tabel [tblorderdetail]



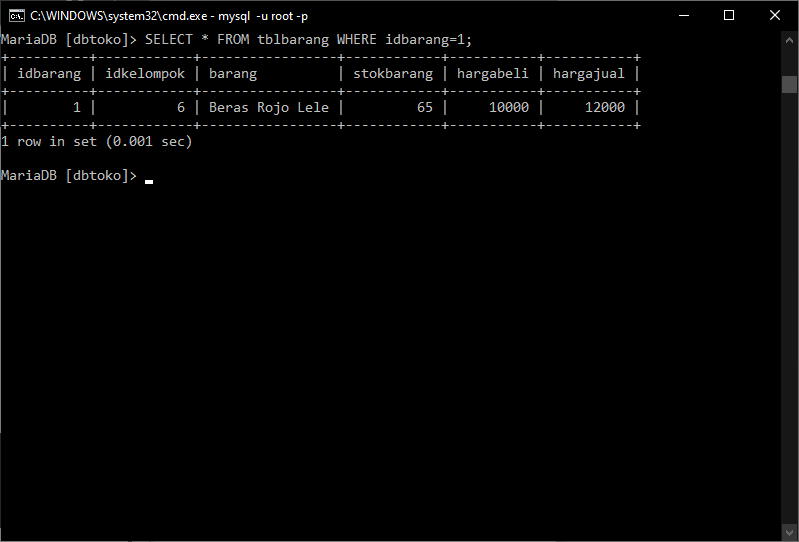
INSERT DATA pada tabel [tblorderdetail]



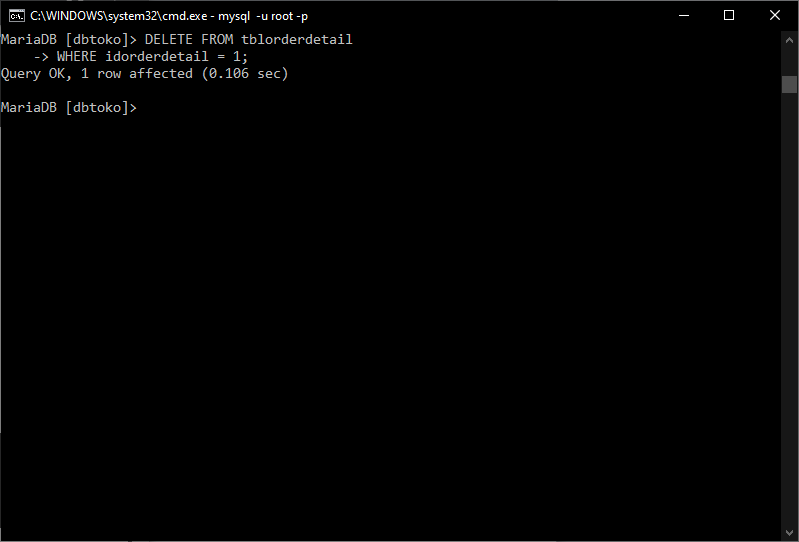
Hasil pada tabel [tblorderdetail]



Hasil pada tabel [tblorder]

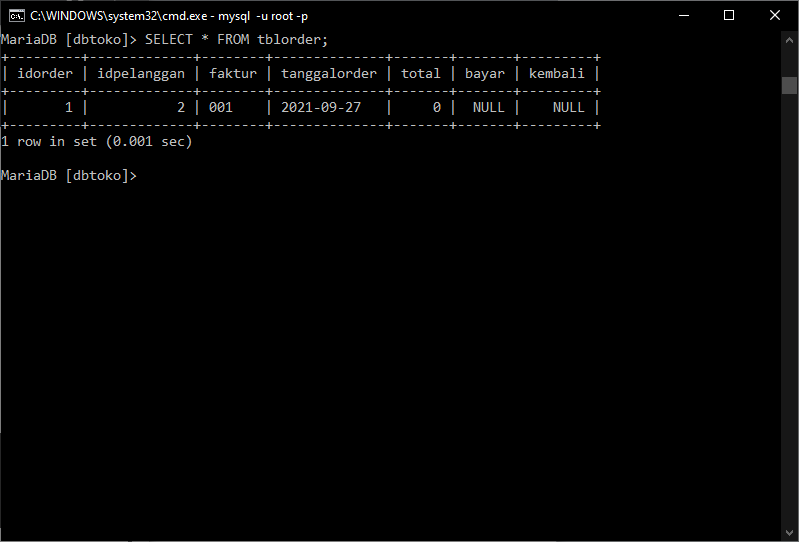


Hasil pada tabel [tblbarang]

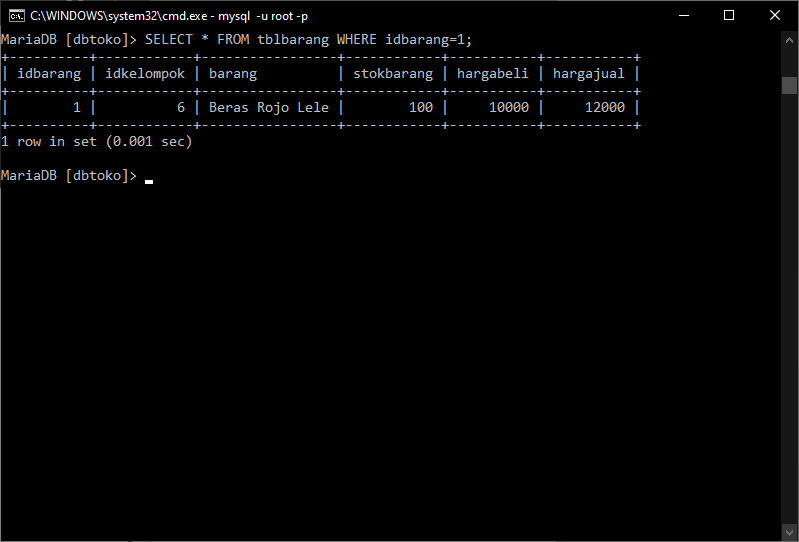


3. PENGUJIAN DELETE PADA TABEL

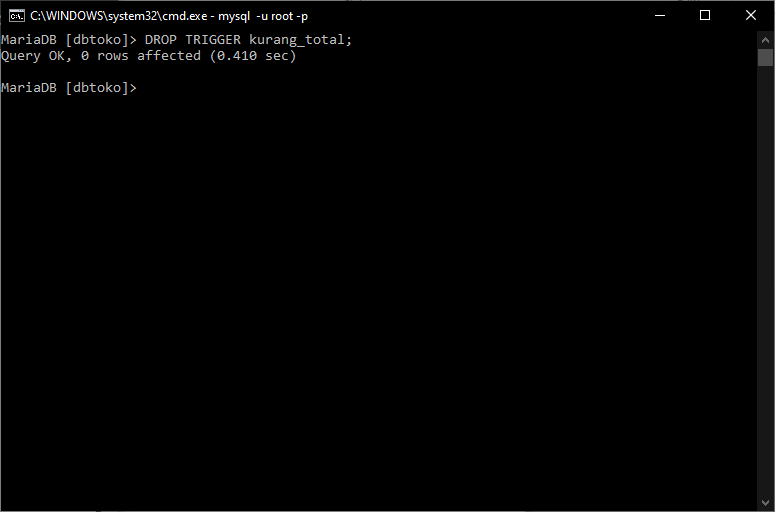
[tblorderdetail] Lakukan DELETE pada tabel [tblorderdetail] dengan perintah sebagai berikut;



Periksa hasil pada tabel [tblorder], setelah dihapus data pada tabel [tblorderdetail] kolom [total] akan kembali kenilai awal

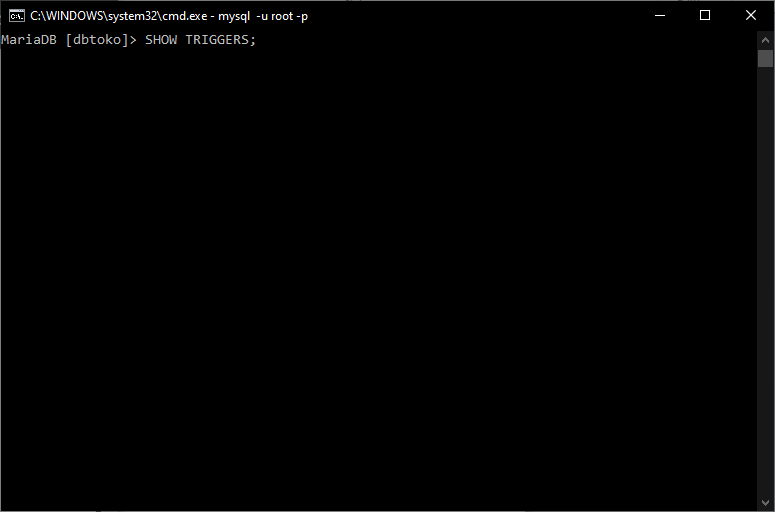


Periksa pada tabel [tblbarang], setelah dihapus data pada tabel [tblorderdetail] kolom [stokbarang] akan kembali kenilai awal

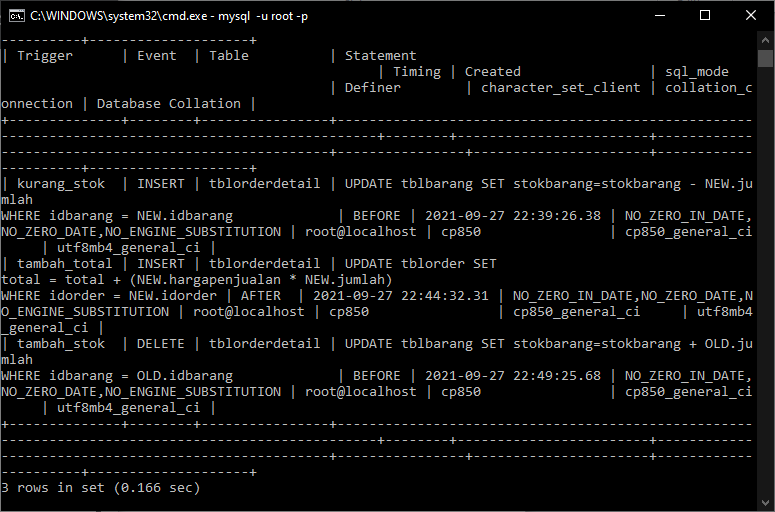


HAPUS TRIGGER

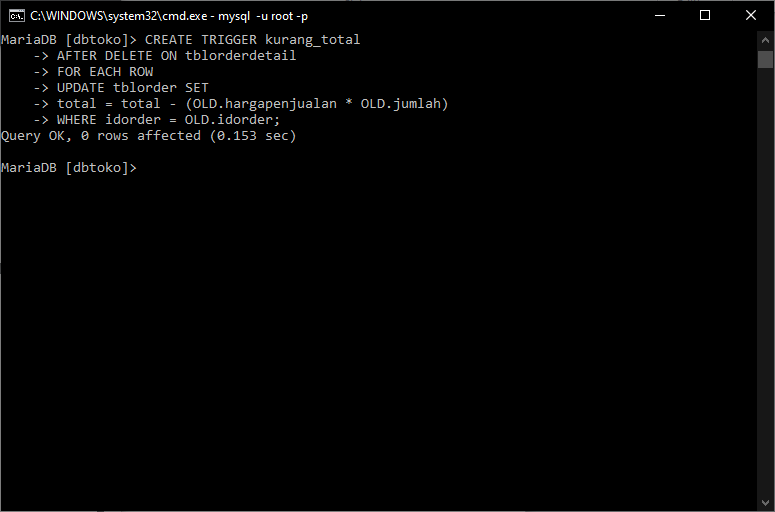
Jika trigger ada yang salah atau tidak digunakan lagi bisa dihapus dengan cara sebagai berikut;



Periksa trigger setelah dihapus;

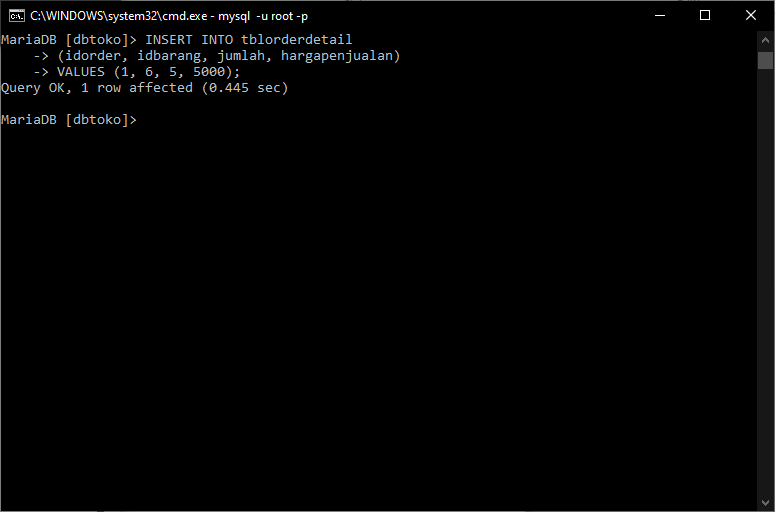


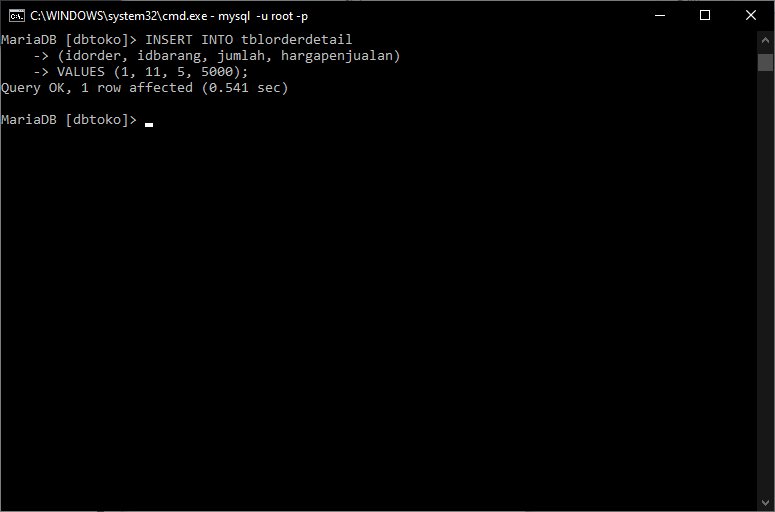
Hasil pemeriksaan



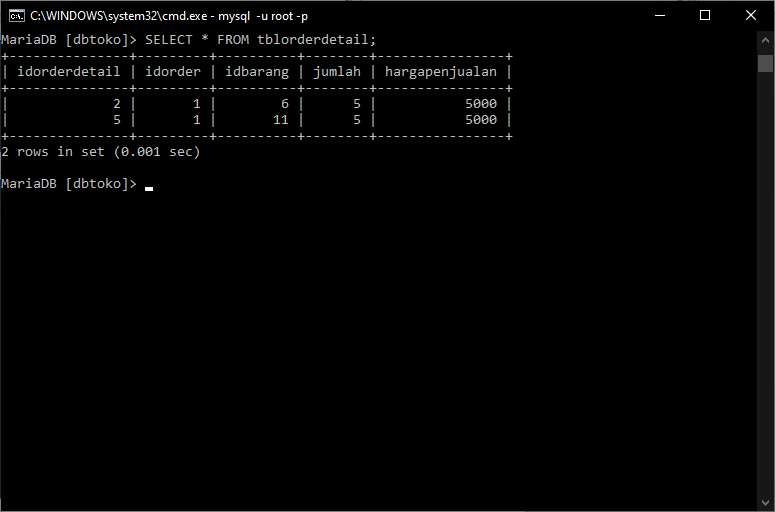
JOIN (GABUNGAN TABEL)

Sebelum belajar JOIN buat kembali Trigger yang telah dihapus pada materi diatas

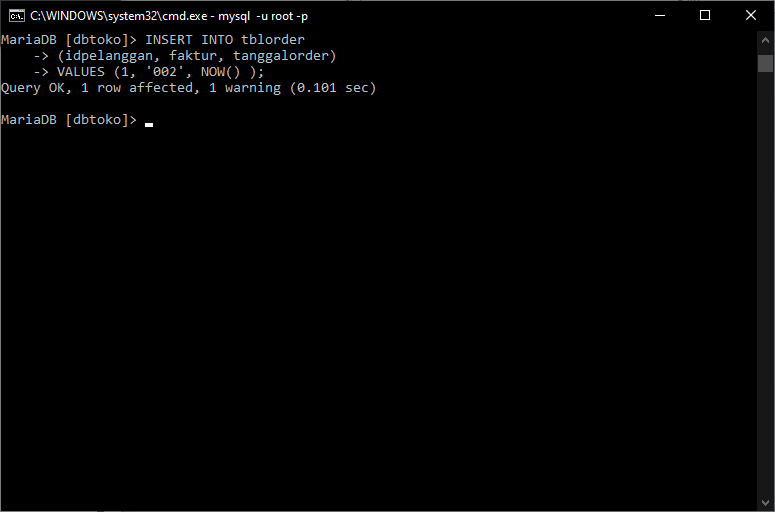




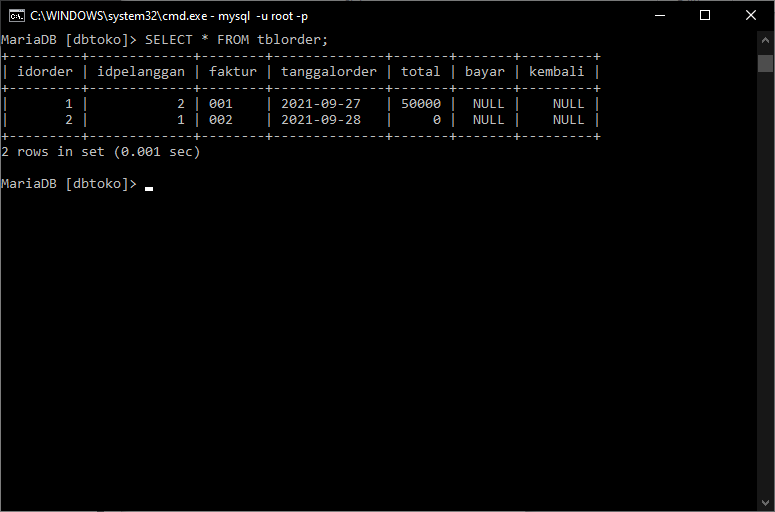
Setelah TRIGGER dibuat kembali lakukan insert data kembali dengan data sebagai berikut;



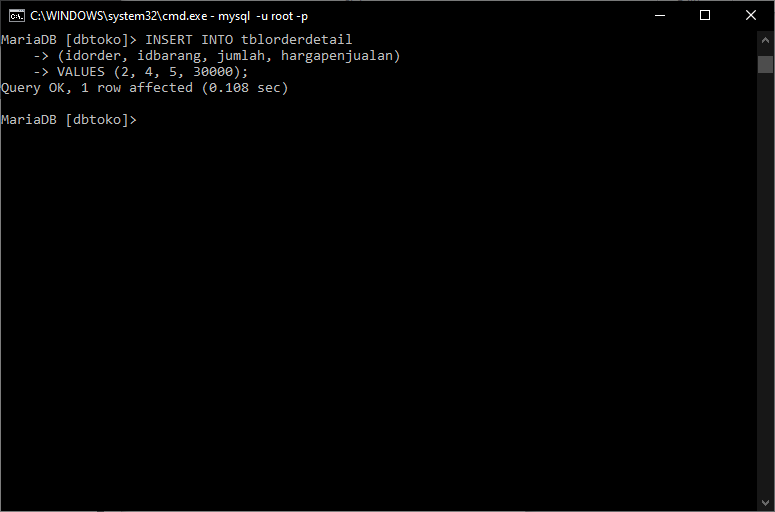
PERIKSA tabel [tblorderdetail]

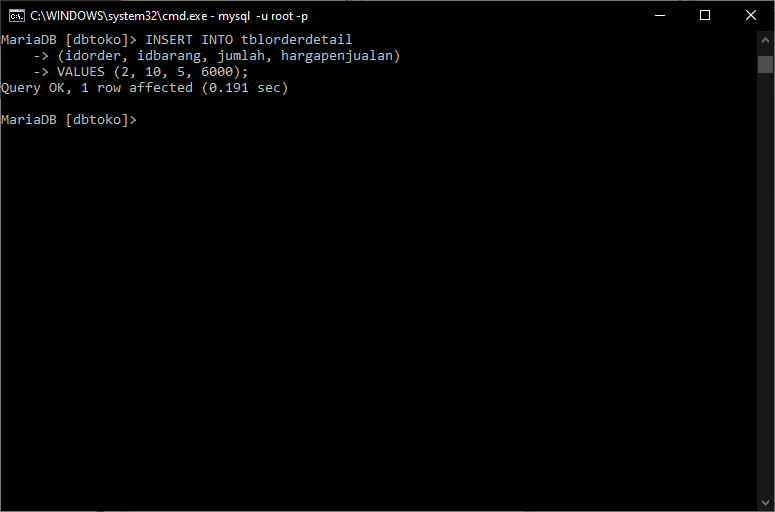


Buat INSERT data pada tabel [tblorder] lagi dengan data sebagai berikut;



Periksa tabel [tblorder]



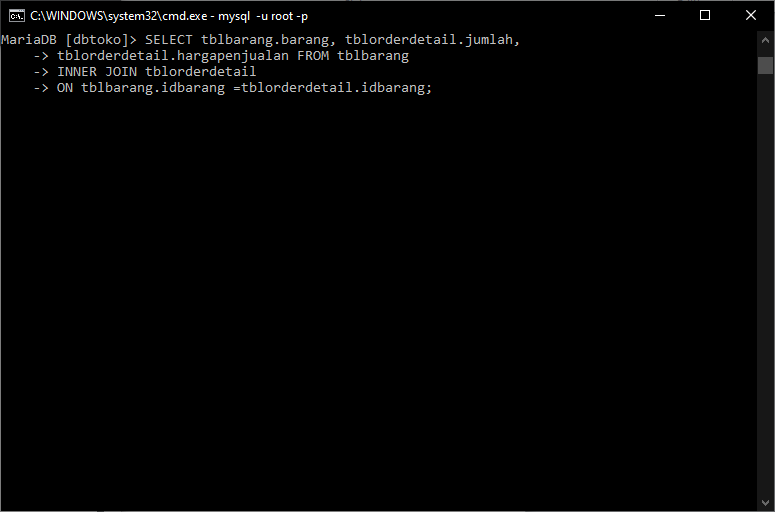


Lakukan insert data pada [tblorderdetail]



PERIKSA tabel [tblorderdetail]

Dari tabel diatas yang masuk ke kolom adalah kode yang sulit dipahami. Untuk memudahkan pembacaan pada tabel maka bisa dibuat gabungan tabel (JOIN) untuk memudahkan pembacaan.



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

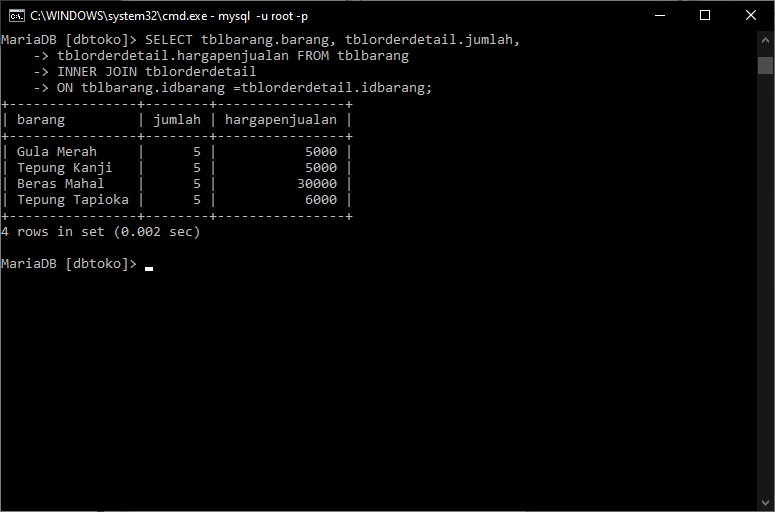
TABEL A = tblbarang

TABEL B = tblorderdetail

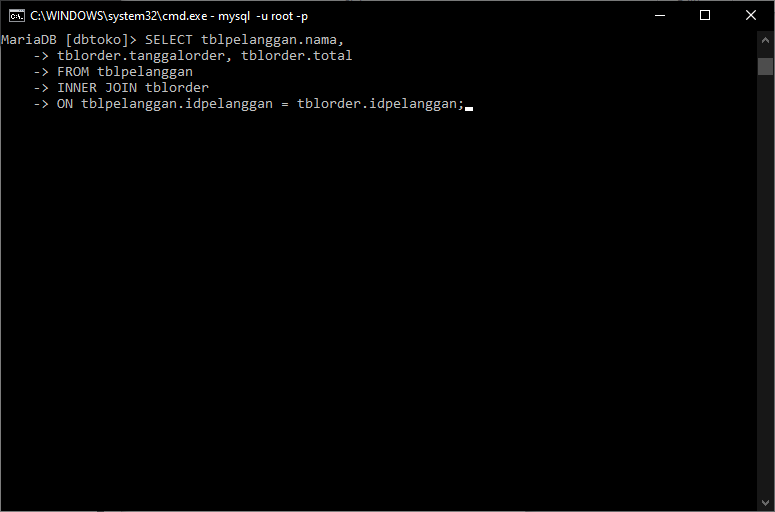
Dari gambar diatas bisa dibuat INNER JOIN untuk mengetahui barang apa yang sudah laku.

SELECT tabel\_master.kolom\_master, tabel\_transaksi.kolom\_transaksi FROM table\_master INNER JOIN tabel\_transaksi ON tabel\_master.kolom\_master = tabel\_transaksi.kolom\_ transaksi;

JOIN HARUS DIMULAI DARI TABEL MASTER



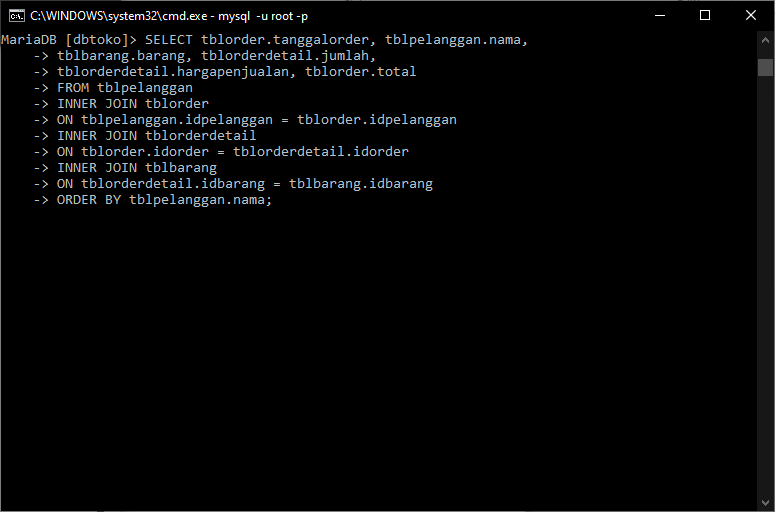
Hasil dari perintah diatas adalah;



INNER JOIN untuk mengetahui pelanggan yang melakukan ORDER

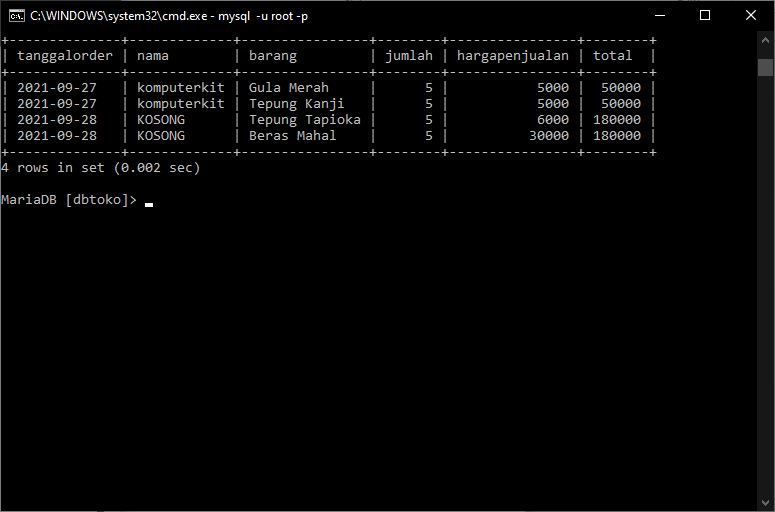


Nama pelanggan yang melakukan order

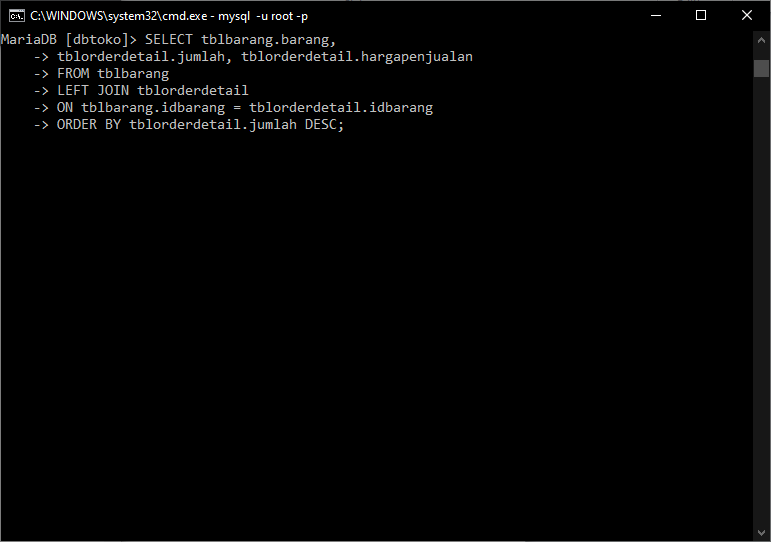


INNER JOIN BANYAK TABEL

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

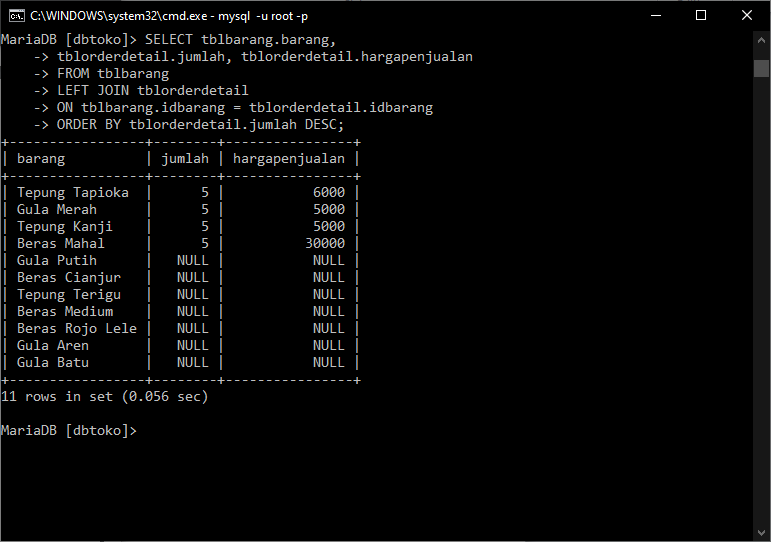


Hasil INNER JOIN dengan banyak tabel

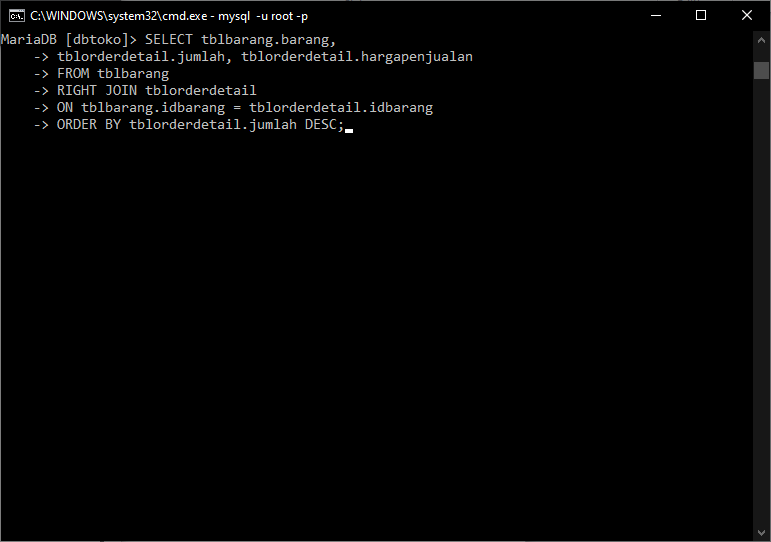


LEFT JOIN (MENAMPILKAN TABEL MASTER)

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



Hasil LEFT JOIN

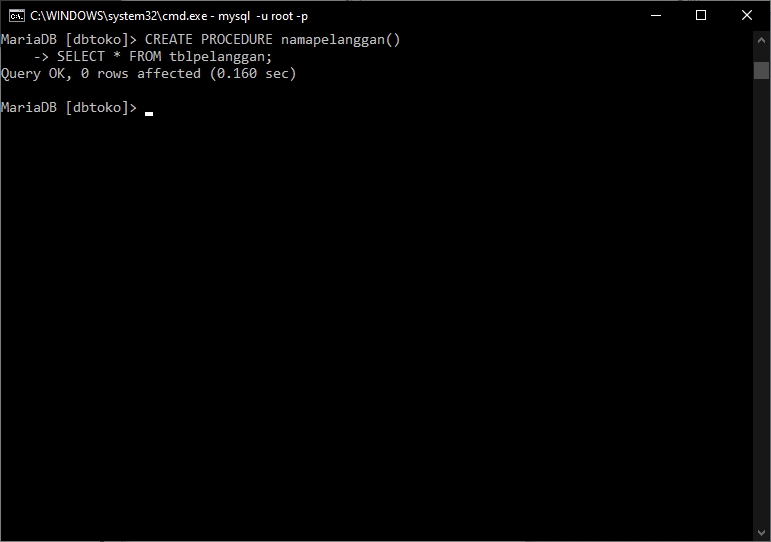


RIGHT JOIN (MENAMPILKAN TABEL TRANSAKSI)

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



Hasil RIGHT JOIN

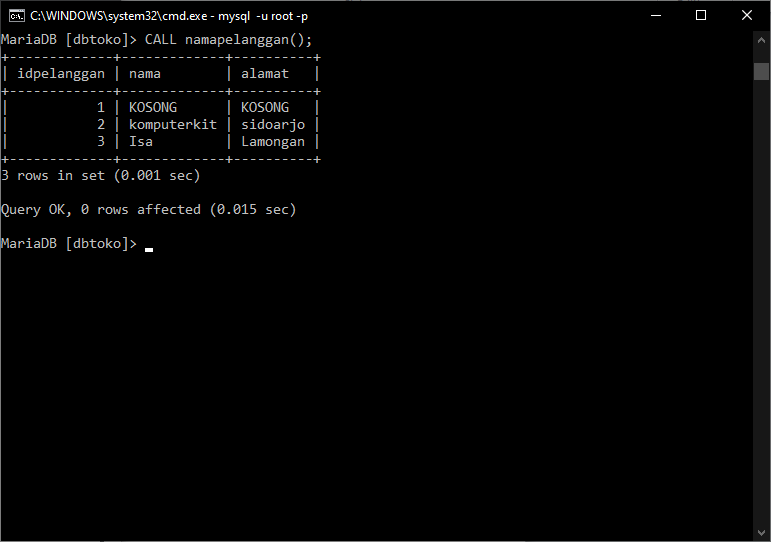


STORE PROCEDURE

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

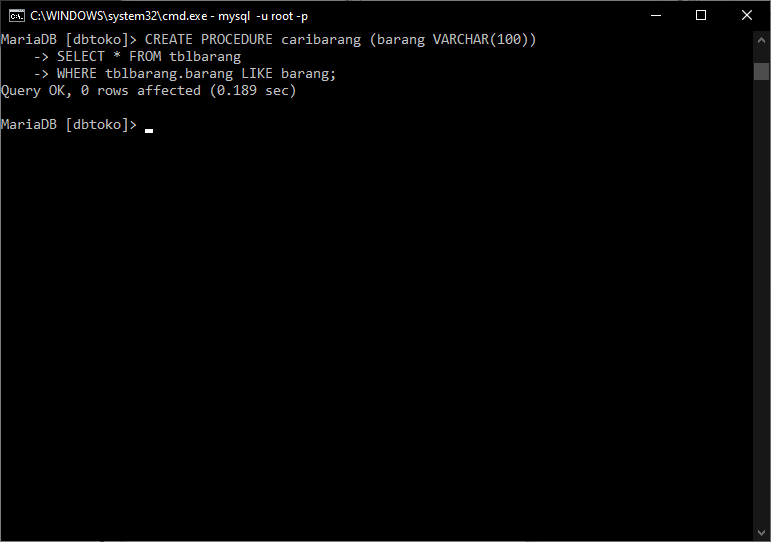
PROCEDURE TANPA PARAMETER

**CREATE PROCEDURE nama\_procedure Isi\_procedure;**

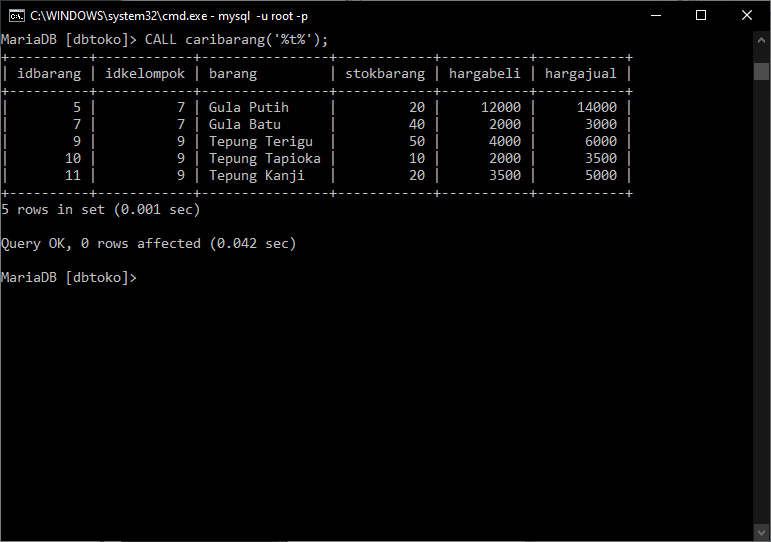


PEMANGGILAN PROCEDURE

CALL nama\_procedure;



PROCEDURE DENGAN PARAMETER



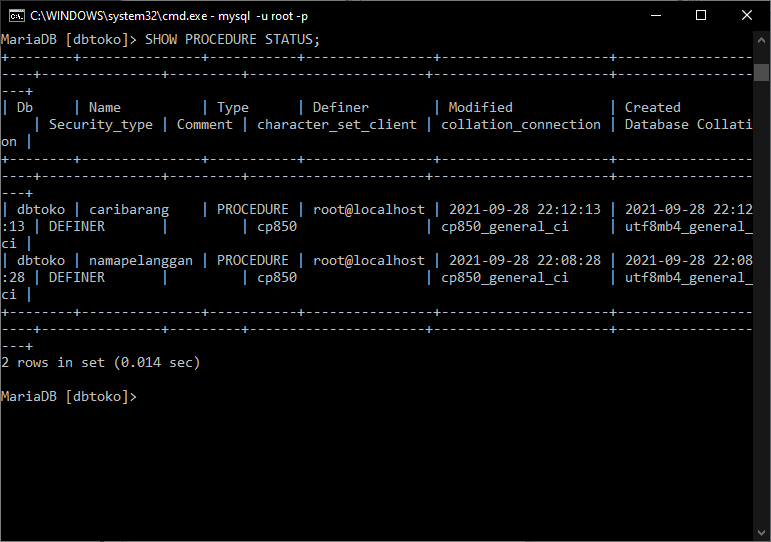
Panggil procedure yang sudah dibuat dengan parameter yang dimasuka [‘%t%’] yang menunjukan nama barang yang mengandung huruf [t]

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

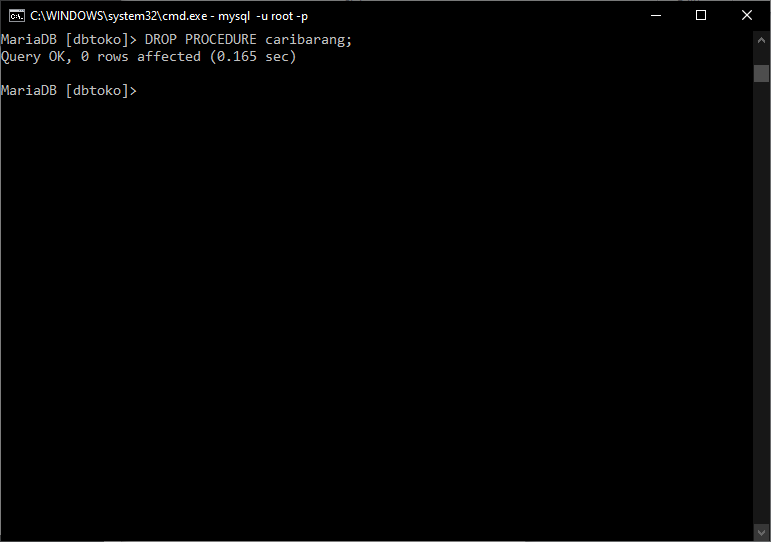


MENAMPILKAN SEMUA PROCEDURE

MySQL bisa menampilkan semua isi procedure dengan perintah berikut;

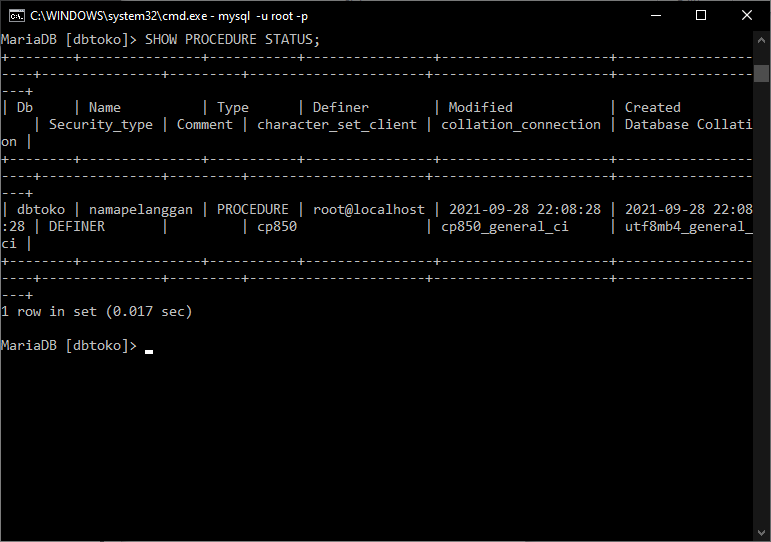


Procedure yang sudah dibuat bisa dilihat pada

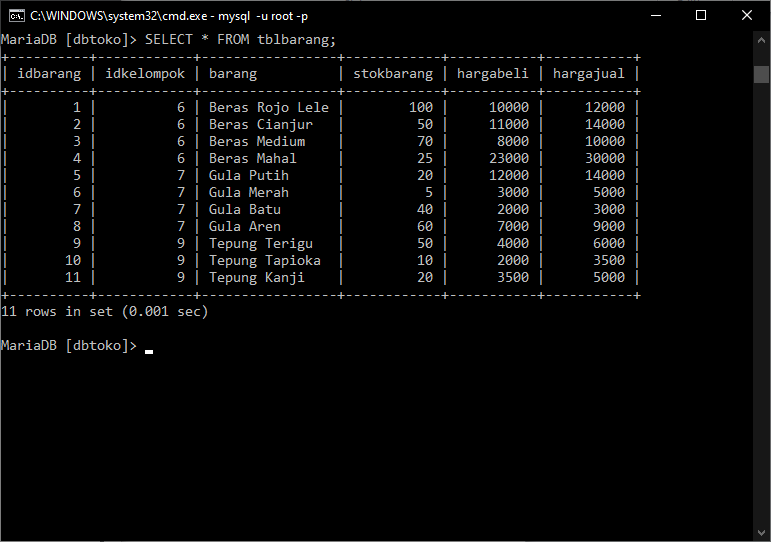


MENGHAPUS STORE PROCEDURE

Untuk menghapus store procedure bisa menggunakan perintah;



Periksa kembali procedure dengan perintah



FUNCTION

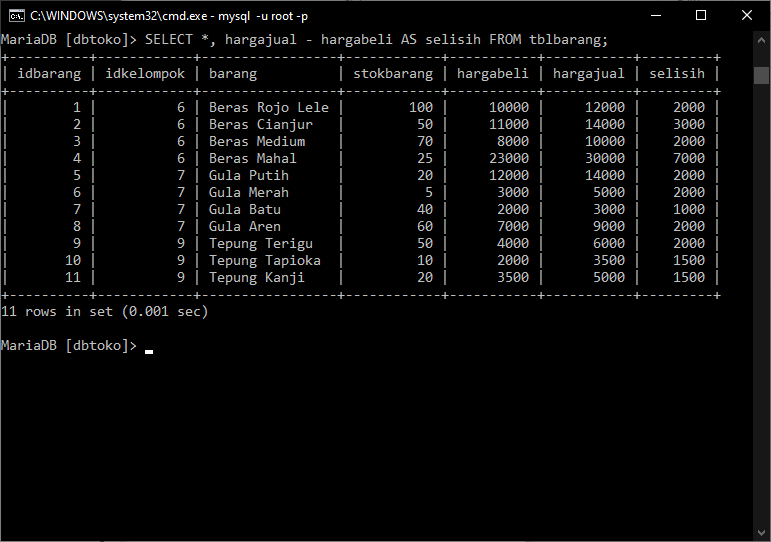
Function adalah blok program yang disimpan di MySQL yang bisa menerima INPUT atau PAREMETER dan MEMILIKI RETURN

MENYIAPKAN ISI FUNCTION

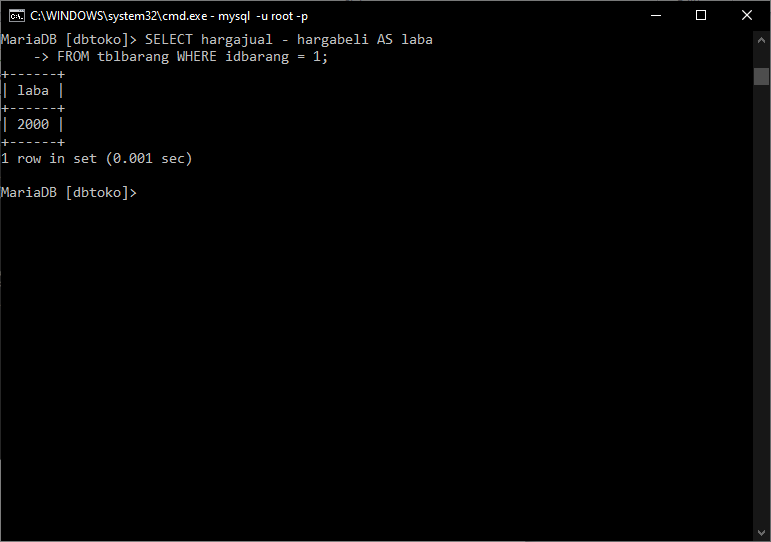
Sebelum membuat Function harus disiapkan dulu code yang akan dibuat sebagai isi function sehingga hasil dari function sesuai dengan yang diharapkan. Pada materi belajar ini kita akan mebuat function untuk menampilkan selisih [hargajual] dan [hargabeli] pada tebel [tblbarang]

Tampilkan tabel [tblbarang]

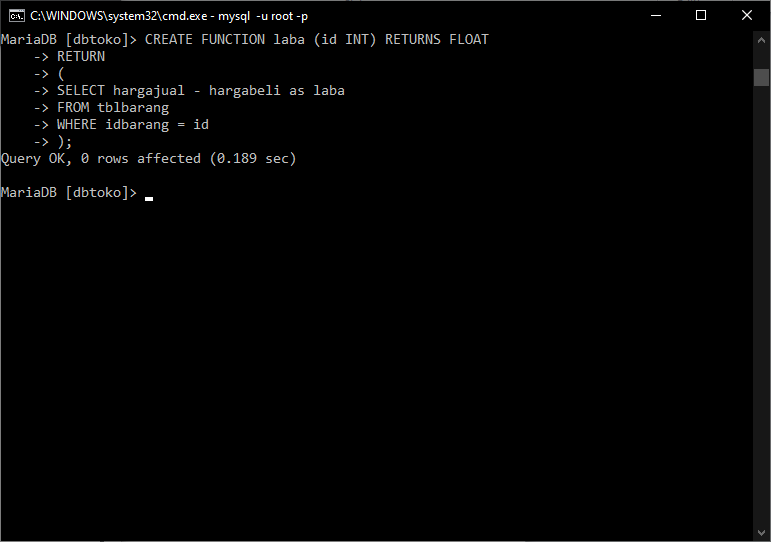
Pada tabel diatas akan ditambahkan 1 kolom lagi sebagai selisih dari [hargajual] dikurangi dengan [hargabeli]



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.

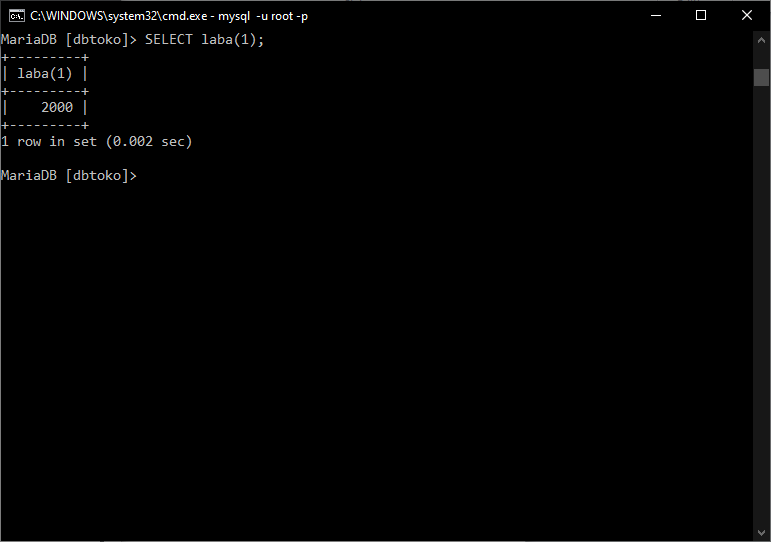


Setelah isi dibuat, saatnya membuat function. Cara membuat function sebagai berikut;



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

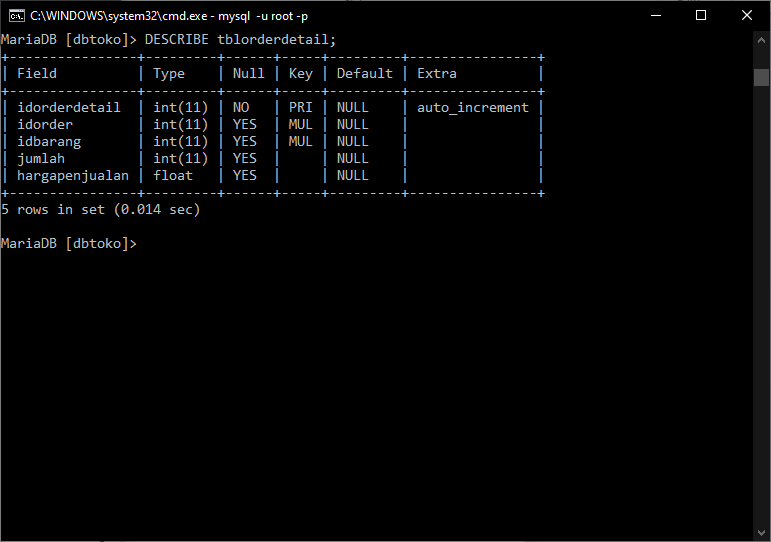
Isi parameter tipe data sesuai dengan data yang dimasukan. Pada contoh diatas parameter yang dimasukan adalah [iDCLrang] dengan tipe data INT. Output yang dihasilkan dari isi function diatas adalah selisih [hargajual] – [hargabeli] yang bertipe data FLOAT;



MEMANGGIL FUNCTION

Untuk memanggil function gunakan perintah berikut;

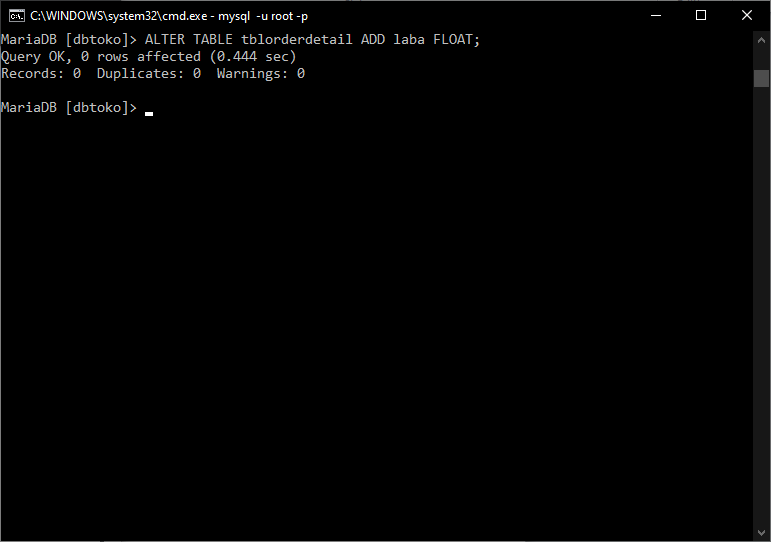
SELECT FUNCTION nama\_function (parameter);



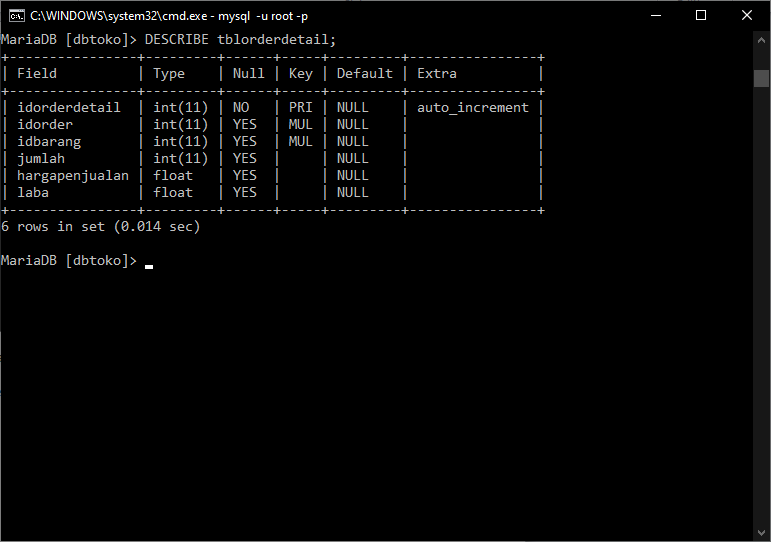
PENGGUNAAN FUNCTION

Materi berikut akan menunjukan penggunaan function pada aplikasi yang sedang dibuat.

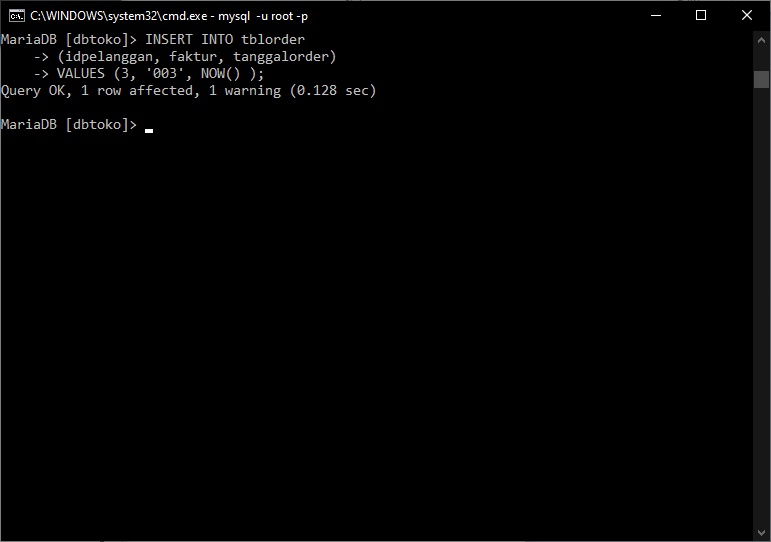
Lihat struktur tabel [tblorderdetail]



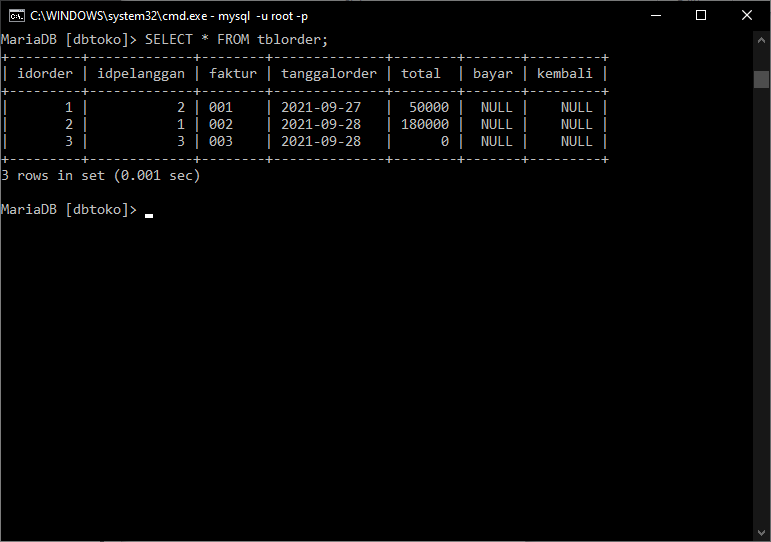
Tambahkan sebuah kolom dengan nama [laba]



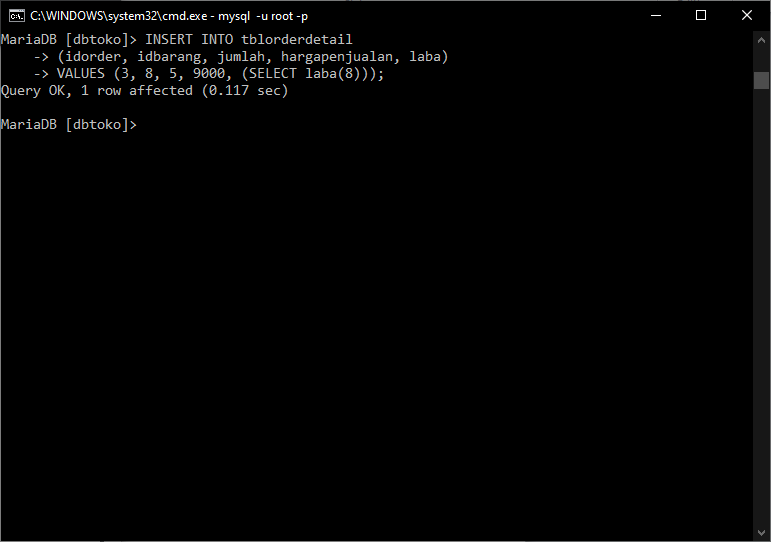
Periksa kembali dengan describe



Lakukan proses order dengan melakukan insert pada tabel [tblorder]

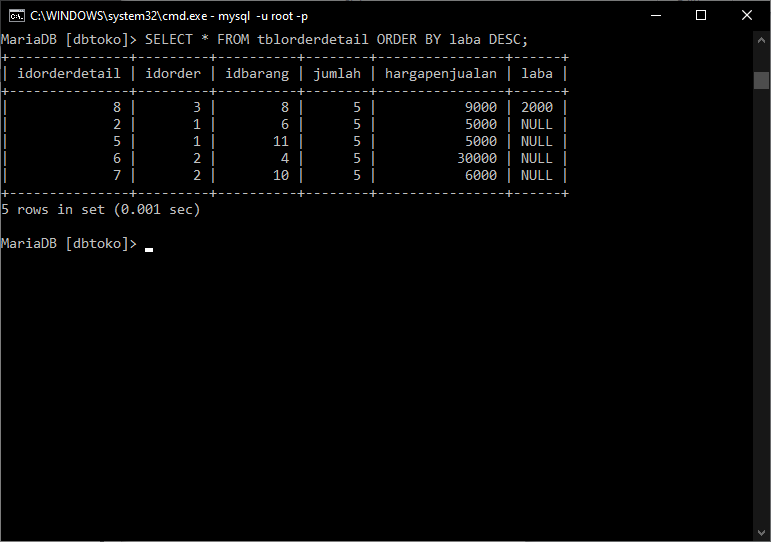


Periksa tabel [tblorder]

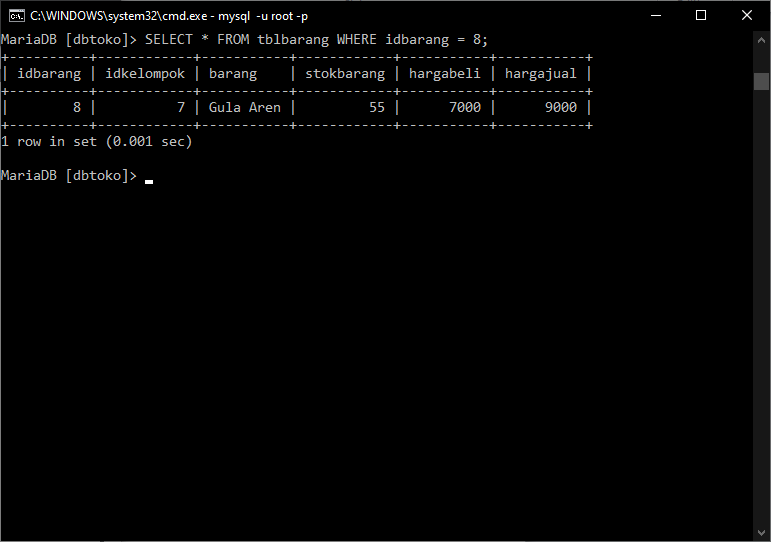


Buat insert pada tabel [tblorderdetail]

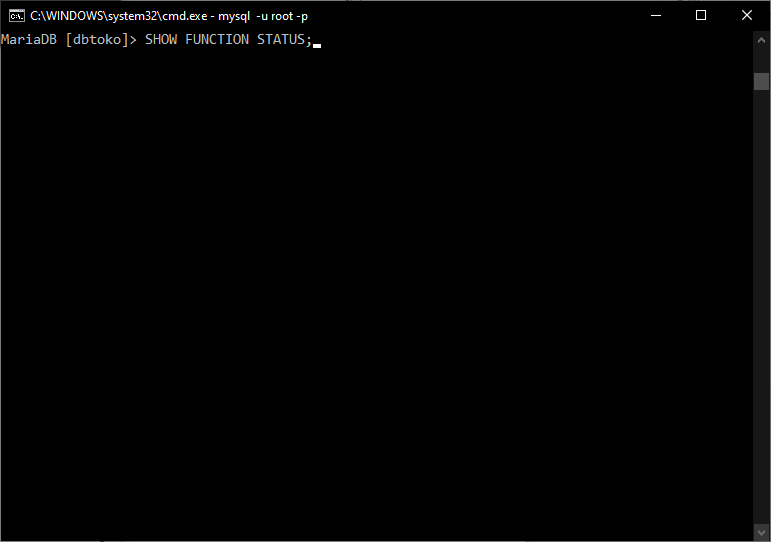
Pada saat INSERT kolom [laba] akan di isi dengan function yang menghitung [hargajual] dikurangi [hargabeli] dimana [iDCLrang] dimasukan sebagai parameter input pada function.



Periksa tabel [tblorderdetail]

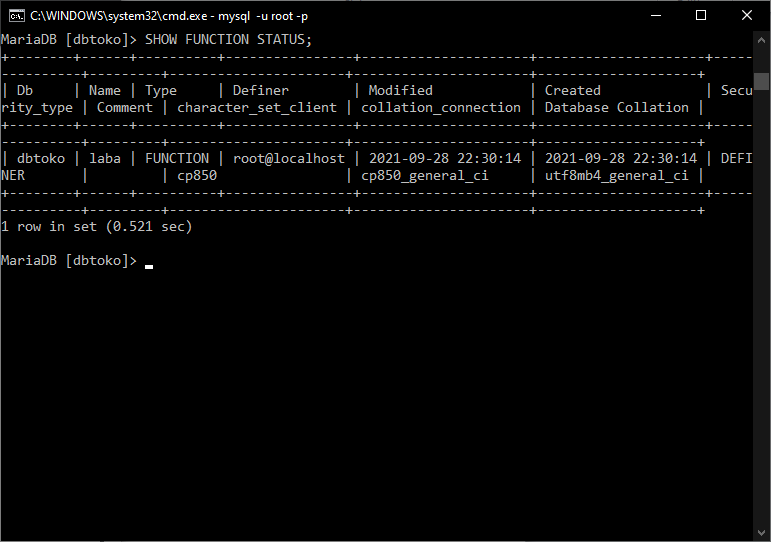


Periksa selisih [hargajual] – [hargabeli] pada tabel [tblbarang]. Jika sesuai berarti function sudah berjalan sesuai dengan yang diharapkan.

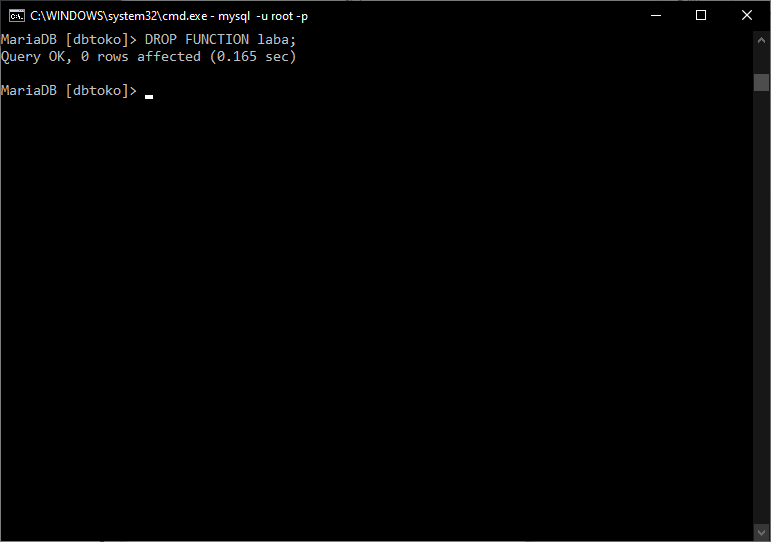


MENAMPILKAN SEMUA FUNCTION

Untuk menampilkan function yang sudah dibuat bisa menggunakan

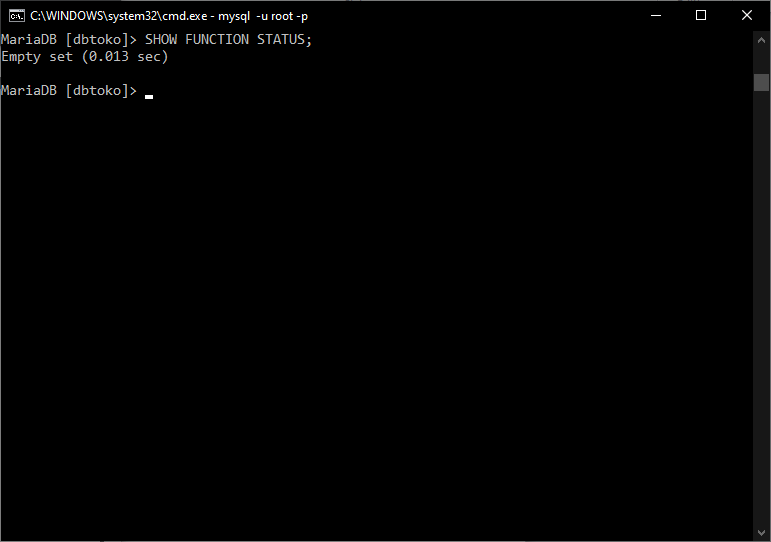


Tampilan function yang telah dibuat

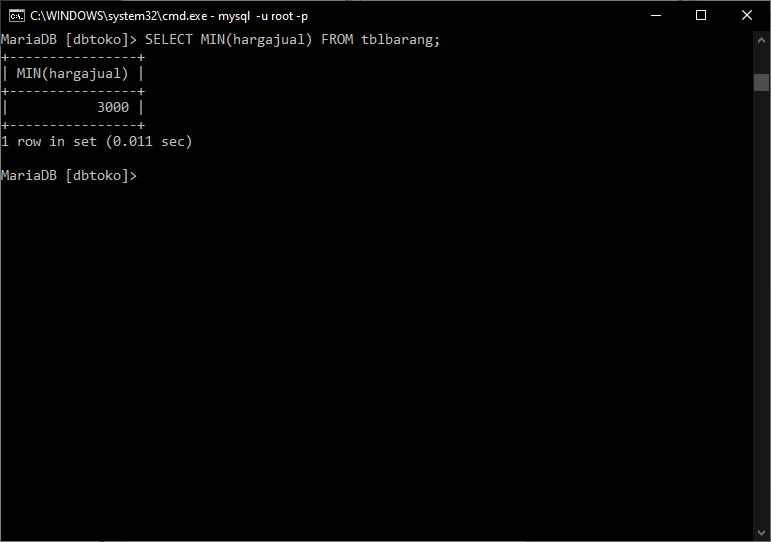


MENGHAPUS FUNCTION

Untuk menghapus function bisa menggunakan perintah berikut;

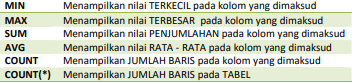


Periksa kembali function yang telah dihapus



SELECT AGGREGATE

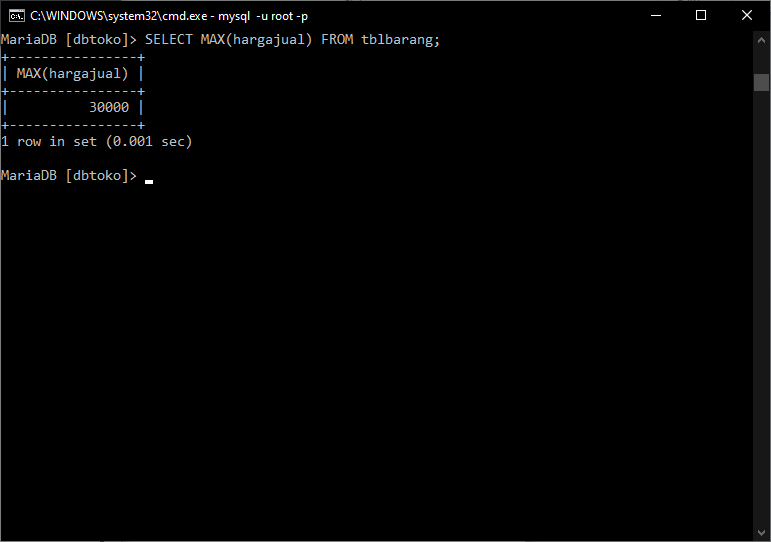
SELECT AGREGATE adalah SELECT yang menampilkan function bawaan dari MySQL. Ada beberapa select aggregate yang akan dipelajari yaitu;



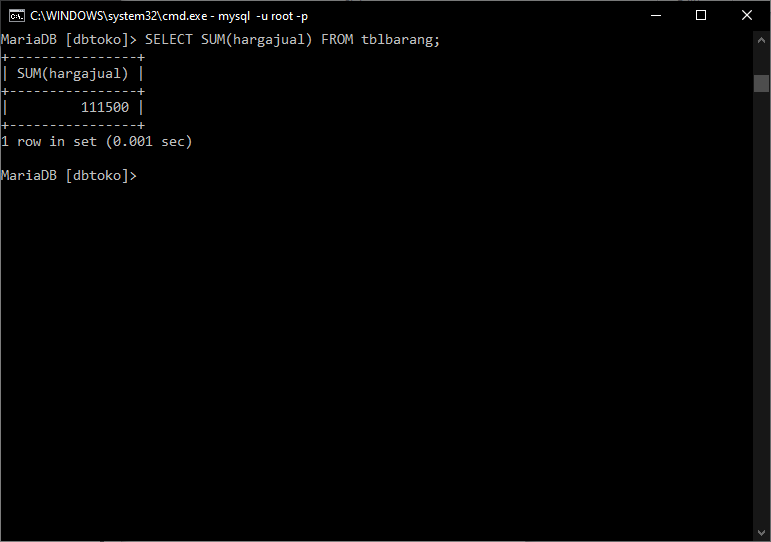
SELECT function\_aggregate(nama\_kolom);

ANTARA NAMA AGGREGATE DAN KURUNG TIDAK BOLEH ADA SPASI

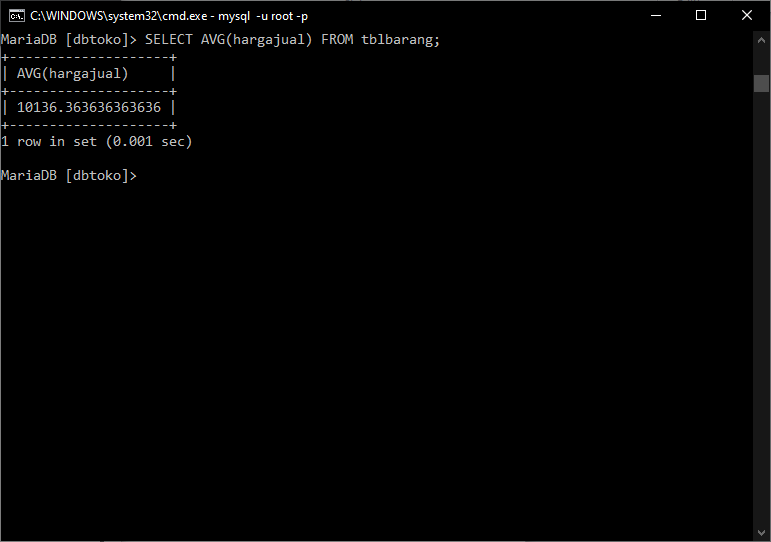
SELECT MIN



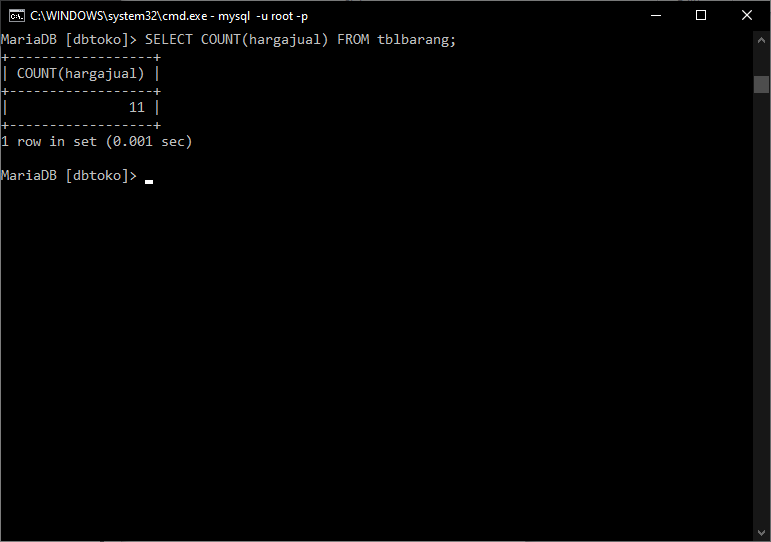
SELECT MAX



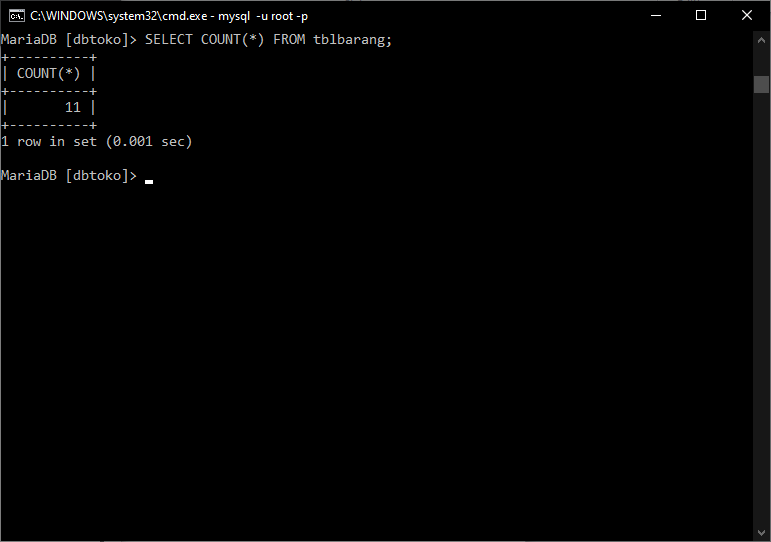
SELECT SUM



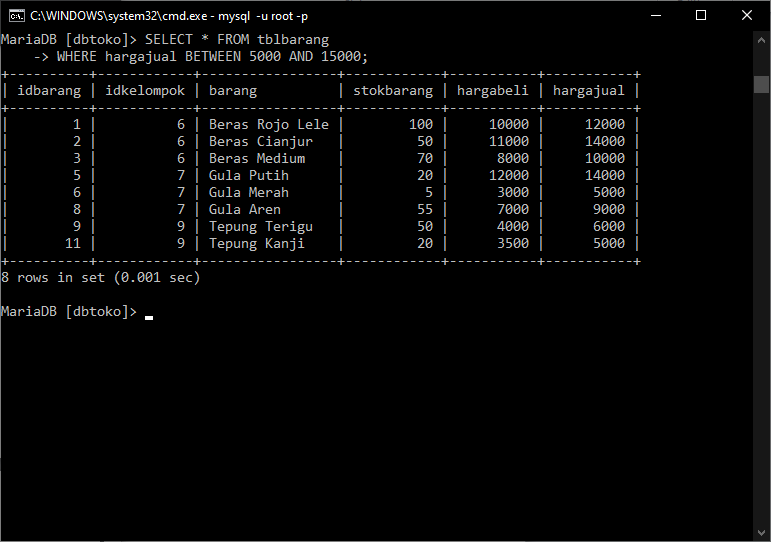
SELECT AVG



SELECT COUNT



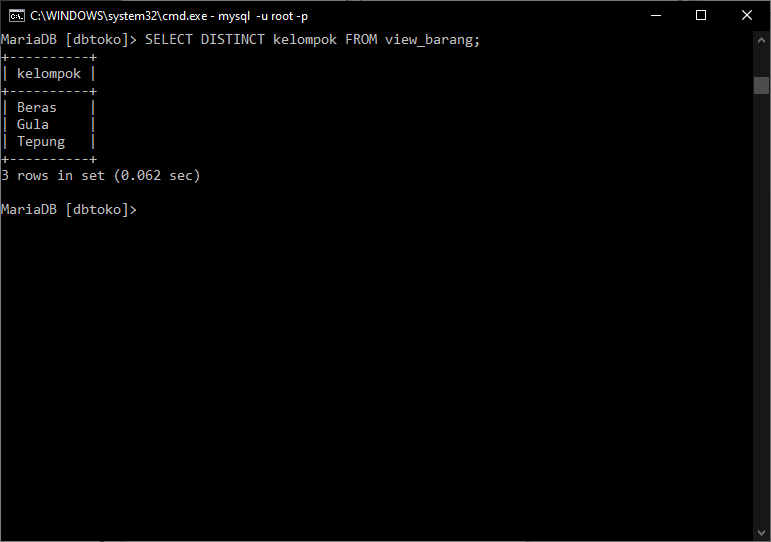
SELECT COUNT(\*)



SELECT BETWEEN (SELECT ANTARA DUA NILAI)

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

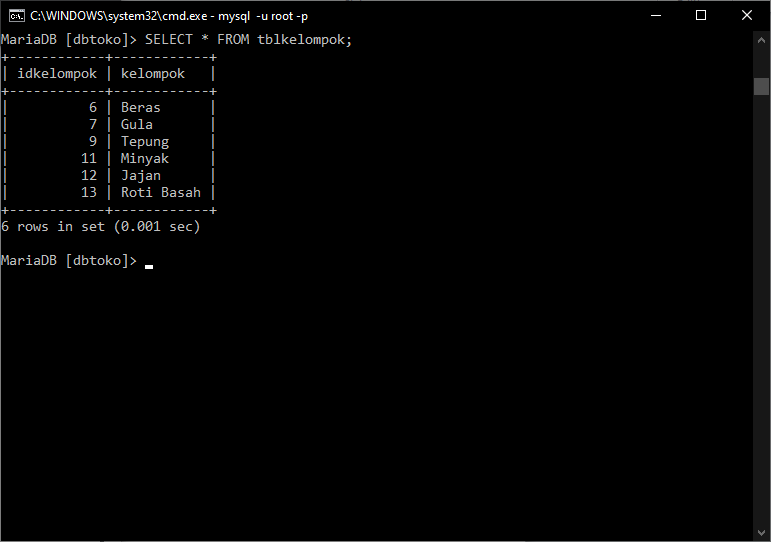
SELECT nama\_kolom\_atau\_\* FROM nama\_tabel WHERE nama\_kolom BETWEEN awal AND akhir



SELECT DISTINCT (MENAMPILKAN DATA YANG SAMA HANYA SATU KALI)

Untuk menampilkan data yang sama satu kali.

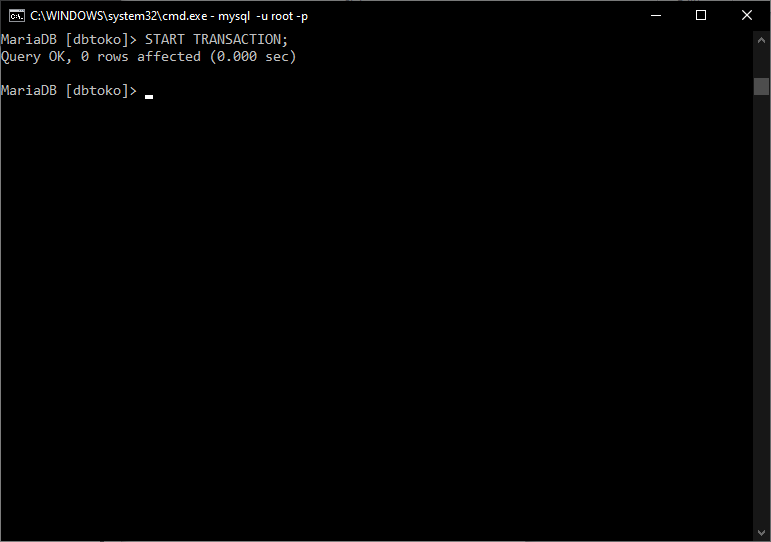
SELECT DISTINCT nama\_kolom FROM nama\_tabel\_view



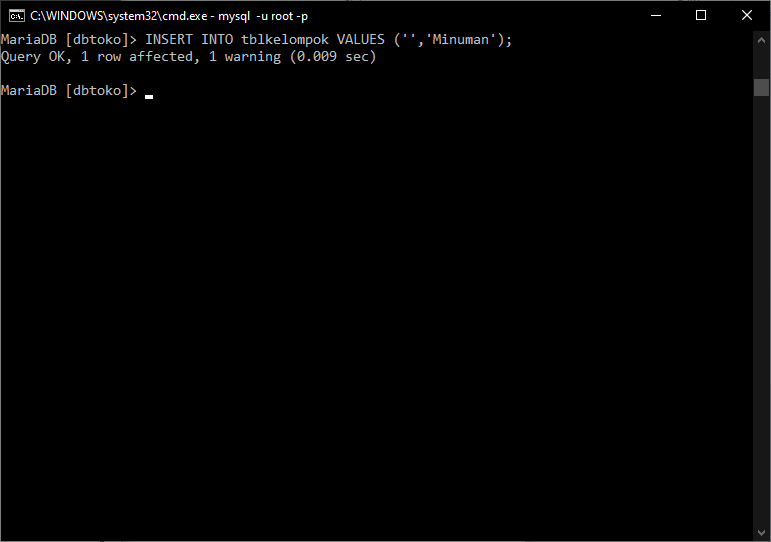
START TRANSACTION, COMMIT, DAN ROLLBACK

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).

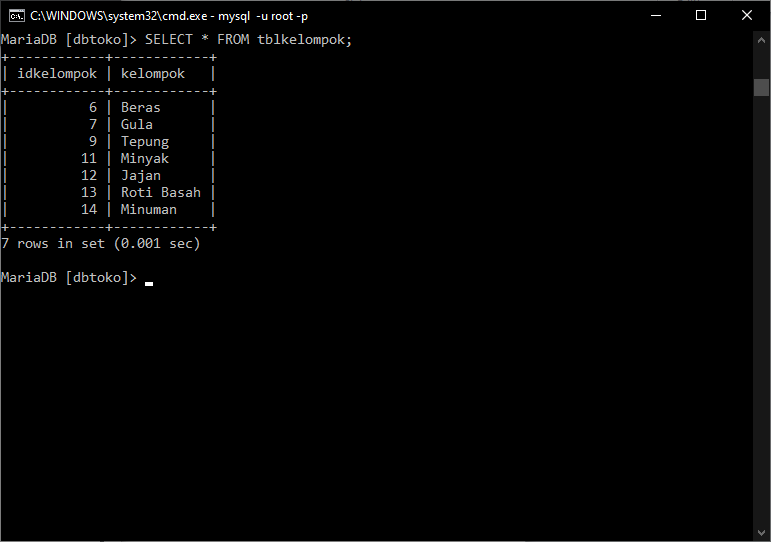
buka kembali tabel [tblkelompok];



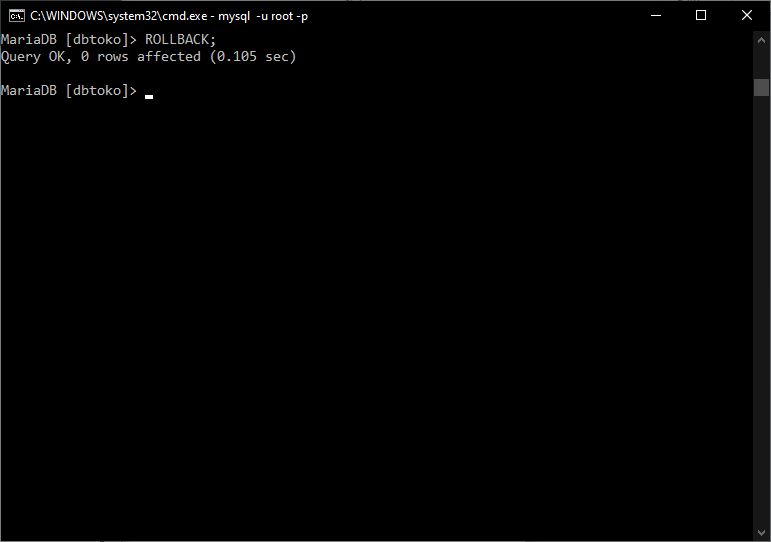
Lakukan perintah berikut, untuk memulai;



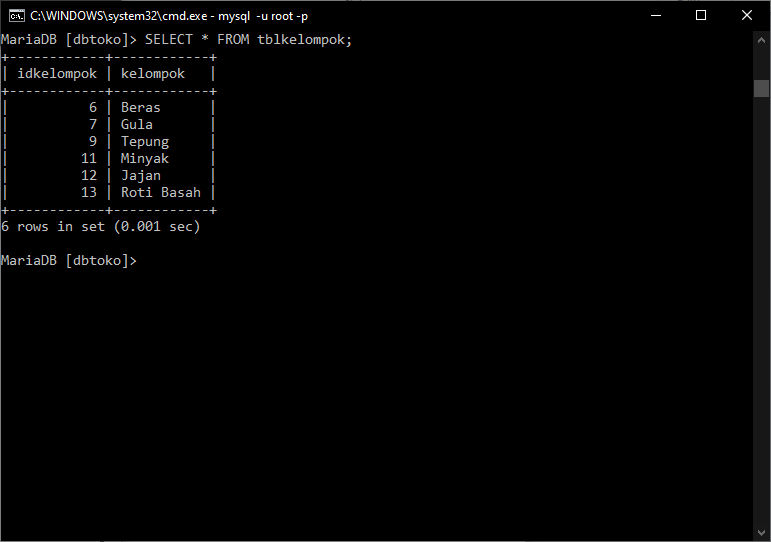
Berikan perintah INSERT sebagai berikut;



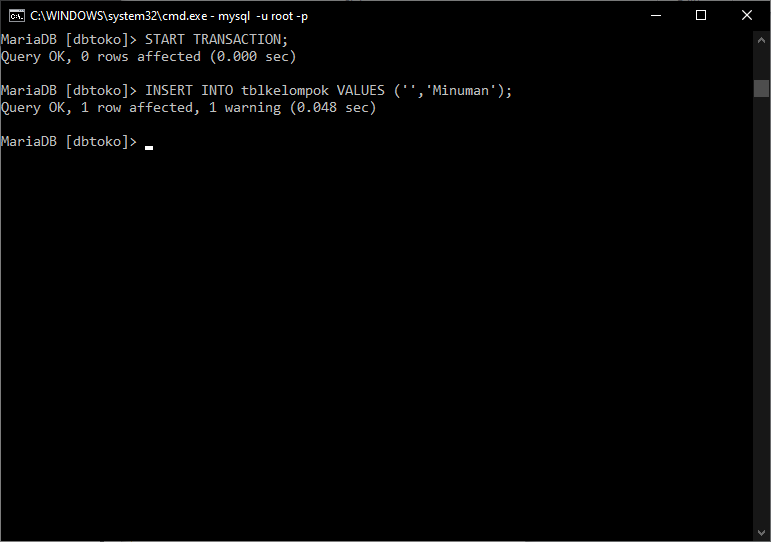
Periksa data yang sudah dimasukan;



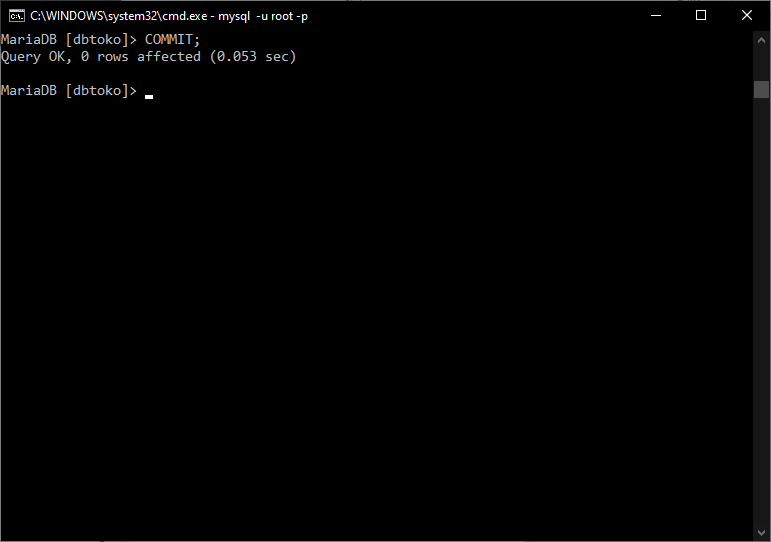
Berikan perintah berikut;



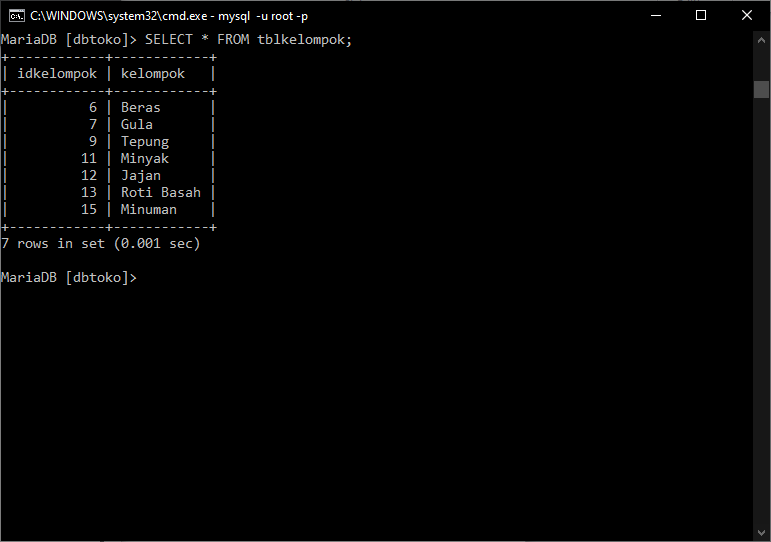
Periksa kembali data yang sudah dimasukan



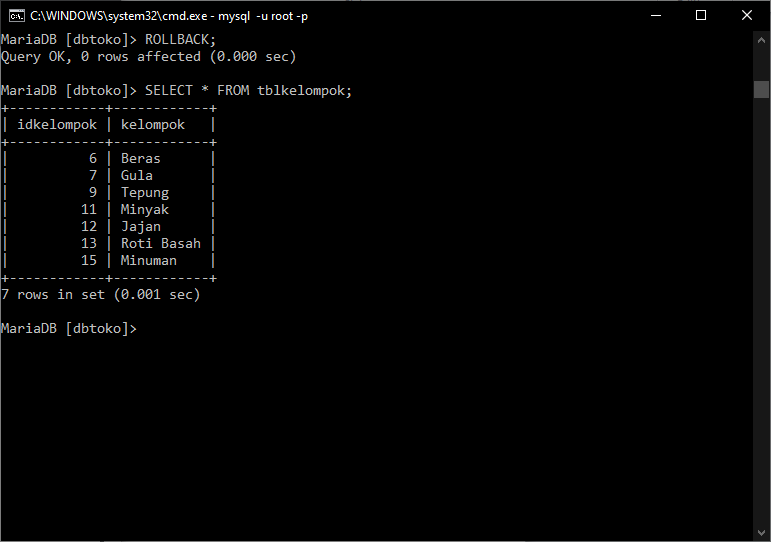
Ulangi perintah INSERT sebelumnya;



Dan ikuti dengan perintah;



Periksa data yang sudah dimasukan;



Lakukan ROLLBACK dan periksa hasilnya. TRANSAKSI YANG SUDAH DI COMMIT TIDAK BISA DI ROLLBACK (undo).

KESIMPULAN START TRANSACTION, COMMIT, & ROLLBACK

• Transaksi adalah perintah (INSERT, UPDATE, DELETE, SELECT)

• Transaksi yang bisa di ROLLBACK (Undo) adalah transaksi yang dimulai dengan START TRANSACTION dan BELUM DI AKHIRI DENGAN COMMIT

• Transaksi yang sudah di COMMIT TIDAK BISA di ROLLBACK

PENGGUNAAN START TRANSACTION, COMMIT, & ROLLBACK

Ketika database yang anda buat sudah digunakan oleh orang yang membeli program ,aka database tersebut akan terisi dengan data VALID (BENAR) sesuai dengan yang dimiliki oleh orang tersebut. Kemudian database tersebut dikembalikan untuk proses modifikasi atau penambahan sesuai dengan kebutuhan orang tersebut. Maka data yang VALID dari database TIDAK BOLEH dicampur dengan DATA DUMMY (data coba - coba) karena anda sedang melakukan PENGUJIAN database. Agar data DUMMY tidak tercampur dengan data VALID gunakan perintah START TRANSACTION sebelum melakukan proses TRANSAKSI. jika pengujian sudah selesai lakukan ROLLBACK. Jangan pernah lakukan COMMIT jika data yang dimasukan data DUMMY.