1.1. Install MySQL

Jika Anda ingin menginstal MySQL di lingkungan Windows, menggunakan installer MySQL adalah cara termudah. Penginstal MySQL memberi Anda wizard yang mudah digunakan yang membantu Anda menginstal MySQL dengan komponen berikut:

* + MySQL Server
  + All Available Connectors
  + MySQL Workbench with Sample Data Models
  + MySQL Notifier
  + Tools for Excel and Microsoft Visual Studio
  + MySQL Sample Databases
  + MySQL Documentation

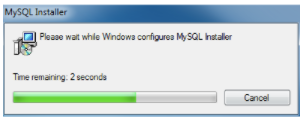
Untuk mengunduh penginstal MySQL, buka tautan berikut http://dev.mysql.com/downloads/installer/. Ada dua file penginstal:

Jika Anda terhubung ke internet saat menginstal MySQL, Anda dapat memilih versi instalasi online mysql-installer-web-community-<version>.exe .

Jika Anda ingin menginstal MySQL secara offline, Anda dapat mengunduh file mysql-installer-community-<version>.exe.

1. 2. Install MySQL via MySQL Installer

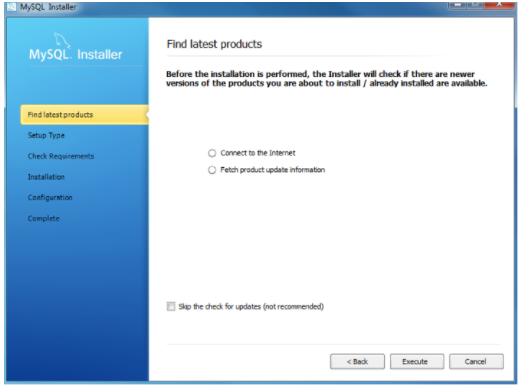
Untuk menginstal MySQL menggunakan penginstal MySQL, klik dua kali pada file penginstal MySQL dan ikuti langkah-langkah di bawah ini:



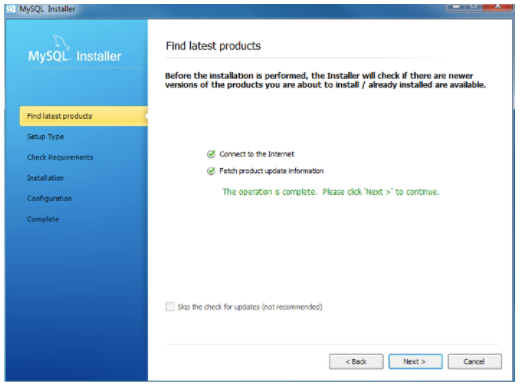
Install MySQL Step 1: Windows configures MySQL Installer



Instal MySQL Langkah 2 – Layar Selamat Datang: Layar pembuka menyediakan beberapa opsi. Pilih opsi pertama: Instal Produk MySQL

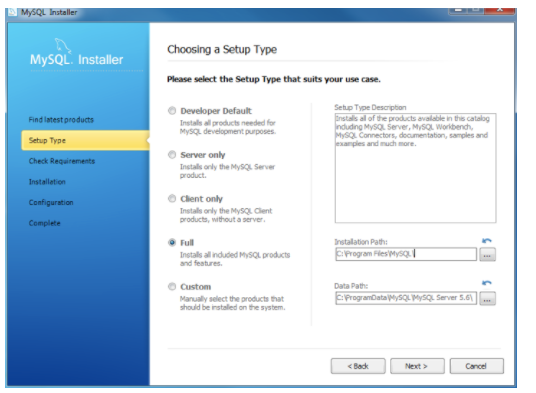


Instal MySQL Langkah 3 – Unduh produk MySQL terbaru: Penginstal MySQL memeriksa dan mengunduh produk MySQL terbaru termasuk server MySQL, MySQL Workbench, dll.

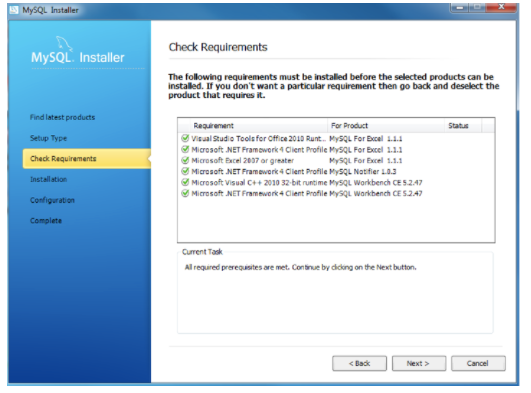


Instal MySQL Langkah 4: Klik tombol Next untuk melanjutkan

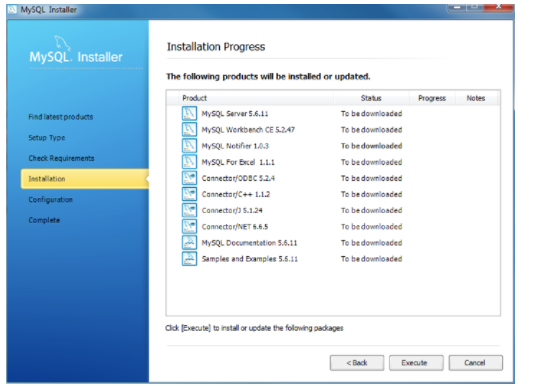
1.3. Install MySQL via MySQL Installer(lanjutan)



Instal MySQL Langkah 5 – Memilih Jenis Pengaturan: ada beberapa jenis pengaturan yang tersedia. Pilih opsi Penuh untuk menginstal semua produk dan fitur MySQL.

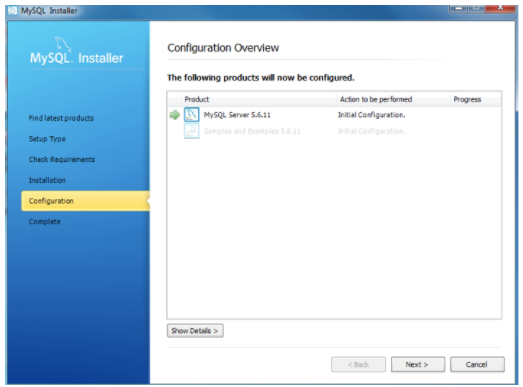


Instal MySQL Langkah 6 – Memeriksa Persyaratan

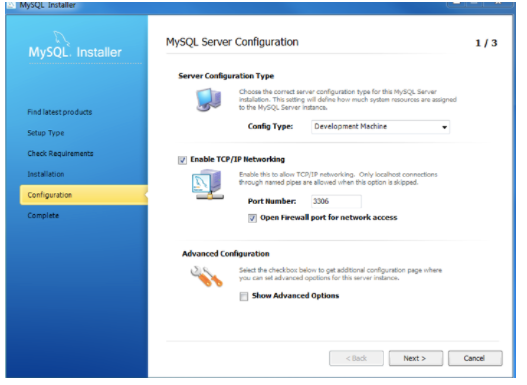


Instal MySQL Langkah 7 – Kemajuan Instalasi: Penginstal MySQL mengunduh semua produk yang dipilih. Ini akan memakan waktu cukup lama, tergantung pada produk yang Anda pilih dan kecepatan koneksi internet Anda.

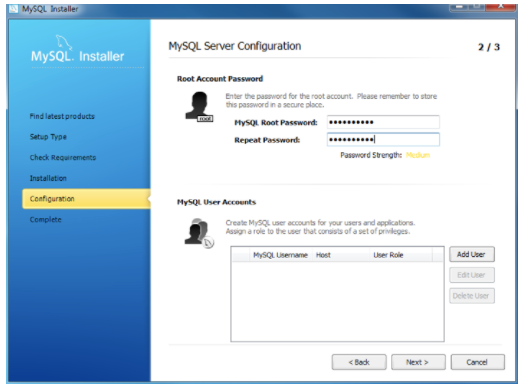
1.4. Configuration Overview(Langkah 8)



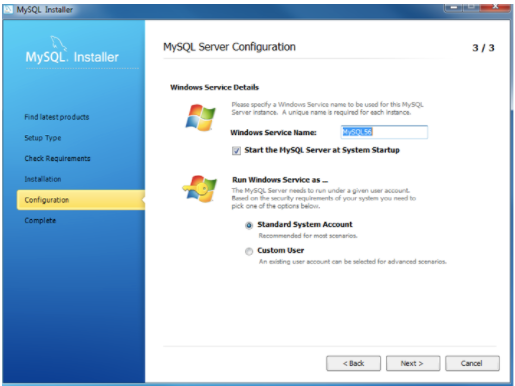
Instal MySQL Langkah 8 – Ikhtisar Konfigurasi. Klik tombol Next untuk mengkonfigurasi Server Database MySQL



Instal MySQL Langkah 8.1 – Konfigurasi Server MySQL: pilih Config Type dan MySQL port (3006 secara default) dan klik tombol Next untuk melanjutkan

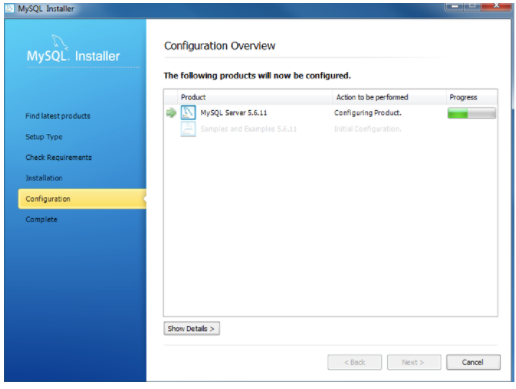


Instal MySQL Langkah 8.2 – Konfigurasi Server MySQL: pilih kata sandi untuk akun root. Harap perhatikan unduhan kata sandi dan simpan dengan aman jika Anda menginstal server database MySQL di server produksi. Jika Anda ingin menambahkan lebih banyak pengguna MySQL, Anda dapat melakukannya di langkah ini.

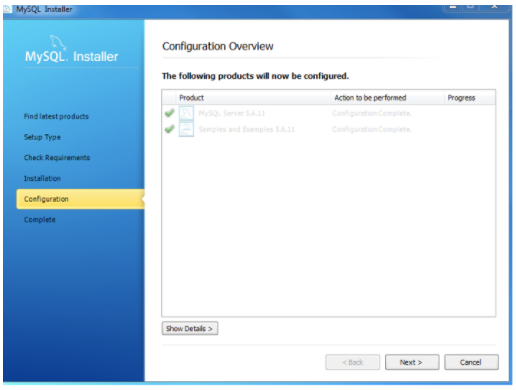


Install MySQL Langkah 8.3 – Konfigurasi Server MySQL: pilih detail layanan Windows termasuk Nama Layanan Windows dan jenis akun, lalu klik tombol Berikutnya untuk melanjutkan.

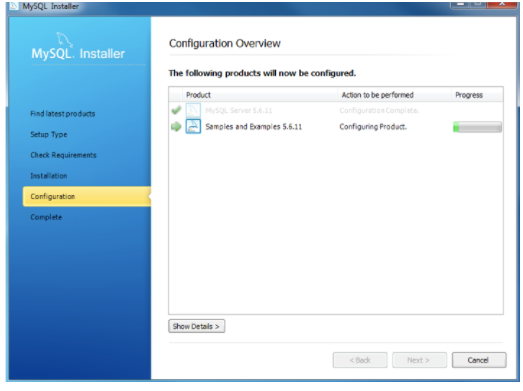
1.5. Conviquration Overview



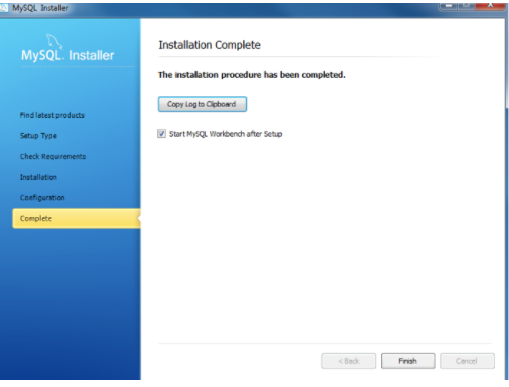
Instal MySQL Langkah 8.1 – Konfigurasi Server MySQL – Sedang Berlangsung: Penginstal MySQL sedang mengonfigurasi server database MySQL. Tunggu sampai selesai dan klik tombol Next untuk melanjutkan.



Instal MySQL Langkah 8.1 – Konfigurasi Server MySQL – Selesai. Klik tombol Berikutnya untuk melanjutkan.



Instal MySQL Langkah 8.2 – Tinjauan Konfigurasi: Penginstal MySQL menginstal database sampel dan model sampel.



Instal MySQL Langkah 9 – Instalasi Selesai: instalasi selesai. Klik tombol Finish untuk menutup wizard instalasi dan meluncurkan MySQL Workbench.

Rangkuman Topik 1

MySQL adalah sistem manajemen basis data relasional (RDBMS) yang banyak digunakan. MySQL gratis dan open source. MySQL sangat ideal untuk aplikasi kecil dan besar.

Siapa yang Menggunakan MySQL?. Situs web besar seperti Facebook, Twitter, Airbnb, Booking.com, Uber, GitHub, YouTube, dll. Sistem Manajemen Konten seperti WordPress, Drupal, Joomla!, Contao, dll. Sejumlah besar pengembang web diseluruh dunia

2.1. Selecting a MySQL database using the mysql client tool

Saat Anda masuk ke server database MySQL menggunakan mysql client  tanpa menentukan nama database, server MySQL akan mengatur database saat ini ke NULL.

Pertama, masuk ke MySQL menggunakan akun pengguna root:

mysql -u root -p

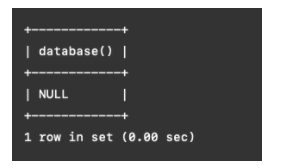
MySQL akan meminta Anda untuk memasukkan kata sandi:

Enter password:

Untuk masuk, Anda harus memberikan kata sandi yang benar dari akun pengguna root dan tekan Enter. Untuk menampilkan database saat ini, Anda menggunakan pernyataan berikut:

SELECT database();

Akan menampilkan berikut ini:



Ini berarti database saat ini tidak diaturl. Jika Anda mengeluarkan pernyataan, MySQL akan mengeluarkan kesalahan. Sebagai contoh:

SELECT \* FROM t;

Error:

ERROR 1046 (3D000): No database selected

Bahasa kode : plaintext (teks biasa)

Untuk memilih database yang akan digunakan, Anda menggunakan pernyataan USE:

USE database\_name;

Misalnya, pernyataan berikut menggunakan pernyataan USE untuk mengatur database saat ini ke classicmodels:

USE classicmodels;

Jika Anda melihat pesan berikut, itu berarti Anda telah berhasil mengubah database menjadi classicmodels:

Database changed

Untuk memverifikasinya, Anda dapat menggunakan pernyataan select database() :

SELECT database();

Ini akan mengembalikan sesuatu seperti:



Jika database classicmodels tidak ada, Anda akan mendapatkan error berikut setelah menjalankan pernyataan USE:

ERROR 1049 (42000): Unknown database 'classicmodels'

Dalam hal ini, Anda perlu menemukan database mana yang tersedia di server Anda dengan menggunakan pernyataan show databases:

SHOW DATABASES;

Outputnya mungkin terlihat seperti berikut:

## 2.2. Memilih database saat login

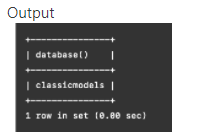
Jika Anda tahu database mana yang ingin Anda gunakan sebelum masuk, Anda bisa menggunakan -D flag . Misalnya, perintah berikut ini terhubung ke database model klasik dengan root akun pengguna:

mysql -u root -D classicmodels -p

Dalam perintah ini, kami menentukan model klasik database setelah flag -D.

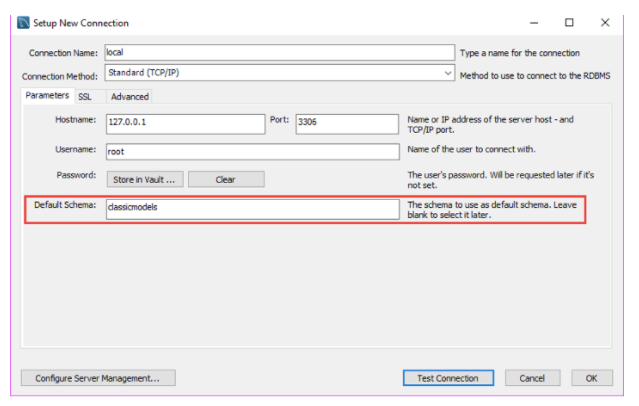
Setelah memasukkan kata sandi dan berhasil masuk, Anda dapat memeriksa database saat ini:

SELECT database();

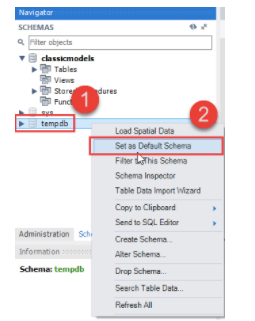
Output  


## 2.3. Memilih database di MySQL Workbench

Jika Anda terhubung ke Server MySQL melalui aplikasi MySQL Workbench, Anda dapat memilih database saat Anda membuat koneksi database seperti yang ditunjukkan pada tangkapan layar berikut:



Setelah login, Anda dapat memilih database lain dengan mengeluarkan pernyataan USE atau menggunakan fitur Set As Default Schema yang disediakan oleh MySQL Workbench:



## 2.4. Pengantar pernyataan MySQL CREATE DATABASE

Untuk membuat database baru di MySQL, Anda menggunakan pernyataan CREATE DATABASE dengan sintaks berikut:

CREATE DATABASE [IF NOT EXISTS] database\_name

[CHARACTER SET charset\_name]

[COLLATE collation\_name]

Dalam sintaks ini:

Pertama, tentukan nama database setelah kata kunci CREATE DATABASE. Nama database harus unik dalam instance server MySQL. Jika Anda mencoba membuat database dengan nama yang sudah ada, MySQL akan mengeluarkan kesalahan.

Kedua, gunakan opsi JIKA TIDAK ADA untuk membuat database secara kondisional jika tidak ada.

Ketiga, tentukan set karakter dan susunan untuk database baru. Jika Anda melewatkan klausa CHARACTER SET dan COLLATE, MySQL akan menjadi default character set dan collation untuk database baru.

**Membuat database baru menggunakan mysql client**

Membuat database baru menggunakan alat klien mysql

Untuk membuat database baru melalui mysql client, Anda mengikuti langkah-langkah berikut:

Pertama, masuk ke Server MySQL menggunakan akun pengguna yang memiliki hak istimewa CREATE DATABASE:

mysql -u root -p

Ini akan meminta Anda untuk memasukkan kata sandi. Untuk mengautentikasi, Anda perlu mengetikkan kata sandi untuk akun pengguna root dan tekan tombol Enter.

Selanjutnya, tampilkan database saat ini yang tersedia di server menggunakan pernyataan SHOW DATABASES. Langkah ini opsional.

SHOW DATABASES;

Hasil



Kemudian, keluarkan perintah CREATE DATABASE dengan nama database mis., testdb dan tekan Enter:

CREATE DATABASE testdb;

Tampilan

Query OK, 1 row affected (0.02 sec)

Setelah itu, gunakan perintah SHOW CREATE DATABASE untuk meninjau database yang dibuat:

SHOW CREATE DATABASE testdb;

MySQL mengembalikan nama database dan set karakter dan susunan database:



Terakhir, pilih database yang baru dibuat untuk digunakan dengan menggunakan pernyataan USE:

USE testdb;

Output

Database changed

Sekarang, Anda dapat mulai membuat tabel dan objek database lainnya di dalam database testdb.

Untuk keluar dari program mysql, ketik perintah exit:

exit

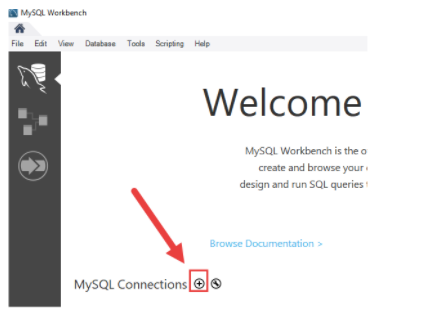
Output:

Bye

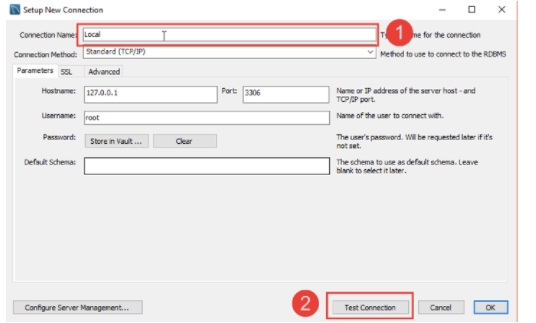
## 2.5. Membuat database baru menggunakan MySQL Workbench

Untuk membuat database baru menggunakan MySQL Workbench, ikuti langkah-langkah berikut:

Pertama, luncurkan MySQL Workbench dan klik tombol setup new connection seperti yang ditunjukkan pada gambar berikut:



Kedua, ketik nama koneksi dan klik tombol **Test Connection.**

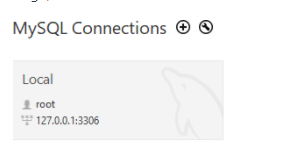


MySQL Workbench menampilkan dialog yang menanyakan kata sandi pengguna root:



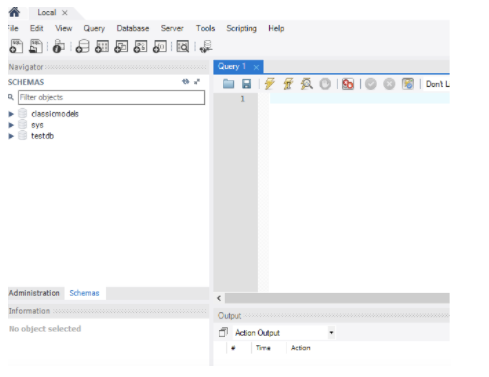
Anda perlu (1) mengetikkan kata sandi untuk pengguna root, (2) memeriksa Simpan kata sandi di brankas, dan (3) klik tombol OK.

Ketiga, klik dua kali nama koneksi Lokal untuk terhubung ke Server MySQL.

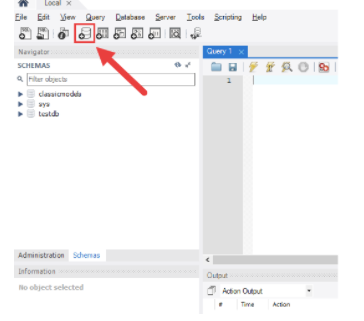


## 2.6. Jendela MySQL Workbench

MySQL Workbench membuka jendela berikut yang terdiri dari empat bagian: Navigator, Query, Information, dan Output.

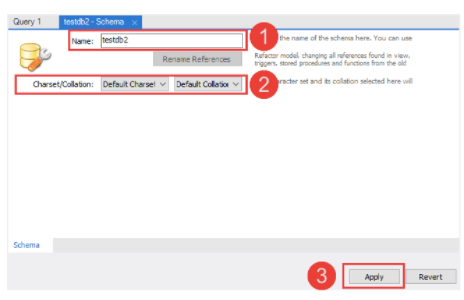


Keempat, klik tombol buat skema baru di server terhubung dari bilah alat:

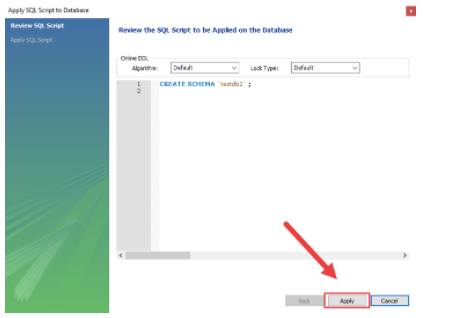


Di MySQL, skema adalah sinonim untuk database. Membuat skema baru juga berarti membuat database baru.

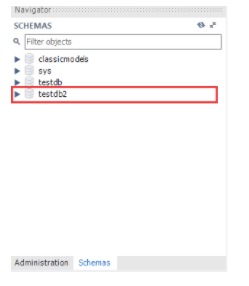
Kelima, jendela berikut terbuka. Anda perlu (1) memasukkan nama skema, (2) mengubah set karakter dan susunan jika perlu, dan klik tombol Terapkan:



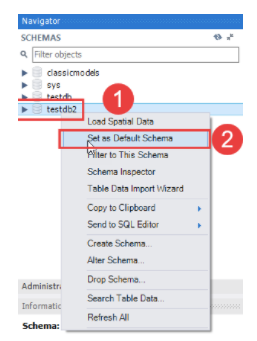
Keenam, MySQL Workbench membuka jendela berikut yang menampilkan skrip SQL yang akan dieksekusi. Perhatikan bahwa perintah pernyataan CREATE SCHEMA memiliki efek yang sama dengan pernyataan CREATE DATABASE.



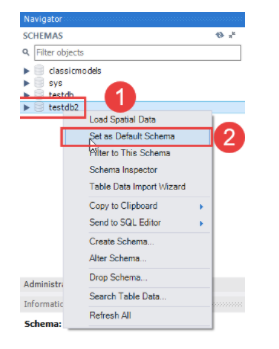
Jika semuanya baik-baik saja, Anda akan melihat database baru dibuat dan ditampilkan di tab skema bagian Navigator.



Ketujuh, untuk memilih database testdb2, (1) klik kanan nama database dan (2) pilih item menu Set as Default Schema:



Node testdb2 terbuka seperti yang ditunjukkan pada tangkapan layar berikut.



## 3.1. Pengantar MySQL DROP DATABASE

Pernyataan DROP DATABASE menghapus semua tabel dalam database dan menghapus database secara permanen. Karena itu,  harus sangat berhati-hati saat menggunakan pernyataan ini.

Berikut ini menunjukkan sintaks dari pernyataan DROP DATABASE:

DROP DATABASE [IF EXISTS] database\_name;

Dalam pernyataan ini, Anda menentukan nama database yang ingin Anda hapus setelah kata kunci DROP DATABASE.

Jika menghapus database yang tidak ada, MySQL akan mengeluarkan kesalahan.

Untuk mencegah kesalahan terjadi jika Anda menghapus database yang tidak ada, Anda dapat menggunakan opsi JIKA ADA. Dalam hal ini, MySQL akan menghentikan pernyataan tanpa mengeluarkan kesalahan apapun.

Pernyataan DROP DATABASE mengembalikan jumlah tabel yang dihapus.

Di MySQL, skema adalah sinonim untuk database. Karena itu, Anda dapat menggunakannya secara bergantian:

DROP SCHEMA [IF EXISTS] database\_name;

## 3.2. MySQL DROP DATABASE menggunakan contoh program mysql

* Pertama, masuk ke Server MySQL menggunakan pengguna root:

mysql -u root -p

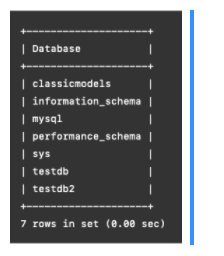
ketik kata sandi untuk pengguna root dan tekan Enter.

Enter password: \*\*\*\*\*\*\*\*

* Kedua, tampilkan semua database menggunakan pernyataan SHOW DATABASES:

SHOW DATABASES;

Output



* Ketiga, hapus database testdb dengan mengeluarkan pernyataan DROP DATABASE:

DROP DATABASE testdb;

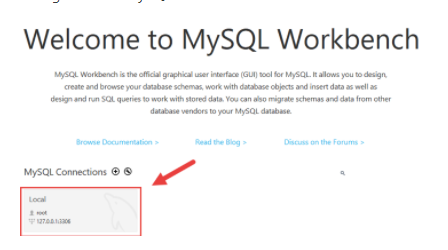
Output

Query OK, 0 rows affected (0.03 sec)

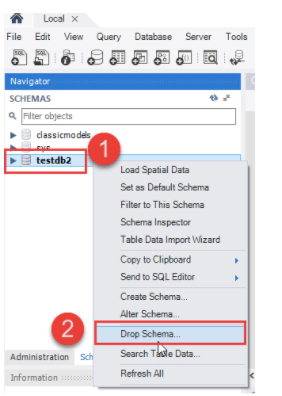
MySQL mengembalikan nol baris yang terpengaruh yang menunjukkan bahwa database testdb tidak memiliki tabel

## 3.3. DROP DATABASE menggunakan MySQL Workbench

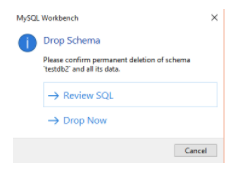
Pertama, luncurkan MySQL workbenchL dan log in to the MySQL Server



Kedua, klik kanan database yang ingin Anda hapus, misalnya testdb2 dan pilih opsi Drop Schema....



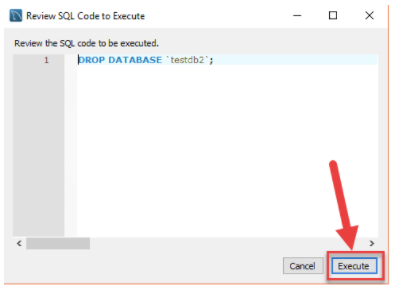
Ketiga, MySQL Workbench menampilkan dialog untuk mengonfirmasi penghapusan.



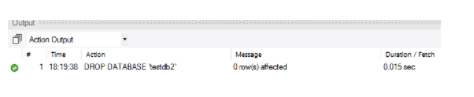
Jika Anda memilih Tinjau SQL, Anda akan melihat pernyataan SQL yang akan dieksekusi. Jika Anda memilih Drop Now, database akan langsung terhapus.

## 3.4. Review SQL

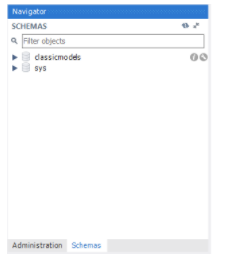
Untuk amannya, mari kita pilih Review SQL:



Keempat, setelah yakin bahwa pernyataan SQL akan menjatuhkan database yang tepat, Anda dapat mengklik tombol Jalankan untuk mengeksekusi pernyataan tersebut.

 MySQL mengembalikan output berikut yang menunjukkan bahwa database berhasil dijatuhkan. Karena testdb2 adalah database kosong, jumlah baris yang terpengaruh adalah nol. 

Jika Anda melihat panel skema, Anda akan melihat bahwa testdb2 tidak ada dalam daftar lagi.



## 3.5. Understanding MySQL Storage Engines

MySQL menyediakan berbagai mesin penyimpanan untuk tabel-tabelnya sebagai berikut:

1. MyISAM
2. InnoDB
3. MERGE
4. MEMORY (HEAP)
5. ARCHIVE
6. CSV
7. FEDERATED

Setiap mesin penyimpanan memiliki kelebihan dan kekurangannya masing-masing. Sangat penting untuk memahami setiap fitur mesin penyimpanan dan memilih yang paling sesuai untuk tabel Anda guna memaksimalkan kinerja database. Pada bagian berikut, kita akan membahas masing-masing mesin penyimpanan dan fitur-fiturnya sehingga Anda dapat memutuskan mana yang akan digunakan.

* MyISAM

MyISAM memperluas mesin penyimpanan ISAM sebelumnya. Tabel MyISAM dioptimalkan untuk kompresi dan kecepatan. Tabel MyISAM juga portabel antara platform dan sistem operasi.

Ukuran tabel MyISAM bisa mencapai 256TB, yang sangat besar. Selain itu, tabel MyISAM dapat dikompresi menjadi tabel hanya-baca untuk menghemat ruang. Saat startup, MySQL memeriksa tabel MyISAM untuk korupsi dan bahkan memperbaikinya jika terjadi kesalahan. Tabel MyISAM tidak aman untuk transaksi.

Sebelum MySQL versi 5.5, MyISAM adalah mesin penyimpanan default saat Anda membuat tabel tanpa menentukan mesin penyimpanan secara eksplisit. Dari versi 5.5, MySQL menggunakan InnoDB sebagai mesin penyimpanan default.

* MERGE

Tabel InnoDB sepenuhnya mendukung kepatuhan ACID dan transaksi. Mereka juga optimal untuk kinerja. Tabel InnoDB mendukung kunci asing, komit, rollback, operasi roll-forward. Ukuran tabel InnoDB bisa mencapai 64 TB.

Seperti MyISAM, tabel InnoDB bersifat portabel antara platform dan sistem operasi yang berbeda. MySQL juga memeriksa dan memperbaiki tabel InnoDB, jika perlu, saat startup.

* MEMORY (HEAP)

Tabel MERGE adalah tabel virtual yang menggabungkan beberapa tabel MyISAM yang memiliki struktur yang mirip dengan satu tabel. Mesin penyimpanan MERGE juga dikenal sebagai mesin MRG\_MyISAM. Tabel MERGE tidak memiliki indeks sendiri; ia menggunakan indeks dari tabel komponen sebagai gantinya.

Menggunakan tabel MERGE, Anda dapat mempercepat kinerja saat menggabungkan beberapa tabel. MySQL hanya memungkinkan Anda untuk melakukan operasi SELECT, DELETE, UPDATE dan INSERT pada tabel MERGE. Jika Anda menggunakan pernyataan DROP TABLE pada tabel MERGE, hanya spesifikasi MERGE yang dihapus. Tabel yang mendasarinya tidak akan terpengaruh.

* Memory

Tabel memori disimpan dalam memori dan menggunakan indeks hash sehingga lebih cepat dari tabel MyISAM. Masa pakai data tabel memori bergantung pada waktu aktif server database. Mesin penyimpanan memori sebelumnya dikenal sebagai HEAP.

* Archive

Mesin penyimpanan arsip memungkinkan Anda untuk menyimpan sejumlah besar catatan, yang untuk tujuan pengarsipan, ke dalam format terkompresi untuk menghemat ruang disk. Mesin penyimpanan arsip memampatkan catatan saat dimasukkan dan mendekompresnya menggunakan perpustakaan zlib saat dibaca.

Tabel arsip hanya mengizinkan pernyataan INSERT dan SELECT. Tabel ARCHIVE tidak mendukung indeks, sehingga diperlukan pemindaian tabel lengkap untuk membaca baris.

* CSV

Mesin penyimpanan CSV menyimpan data dalam format file comma-separated values ​​(CSV). Tabel CSV menghadirkan cara mudah untuk memigrasikan data ke aplikasi non-SQL seperti perangkat lunak spreadsheet.

Tabel CSV tidak mendukung tipe data NULL. Selain itu, operasi baca memerlukan pemindaian tabel penuh.

* FEDERATED

Mesin penyimpanan FEDERATED memungkinkan Anda untuk mengelola data dari server MySQL jarak jauh tanpa menggunakan cluster atau teknologi replikasi. Tabel federasi lokal tidak menyimpan data. Saat Anda meminta data dari tabel gabungan lokal, data ditarik secara otomatis dari tabel gabungan jarak jauh.

## Rangkuman Topik 3

Berbagai mesin penyimpanan MySQL atau jenis tabel. Sangat penting untuk memahami fitur dari setiap mesin penyimpanan di MySQL sehingga dapat menggunakannya secara efektif untuk memaksimalkan kinerja database. Gunakan pernyataan MySQL DROP DATABASE untuk menghapus database yang ada di server.

## 4.1. MySQL CREATE TABLE syntax

pernyataan CREATE TABLE  untuk membuat tabel baru dalam database.

Berikut ini mengilustrasikan sintaks dasar dari pernyataan CREATE TABLE:

CREATE TABLE [IF NOT EXISTS] table\_name(

column\_1\_definition,

column\_2\_definition,

...,

table\_constraints

) ENGINE=storage\_engine;

**Pertama**, a tentukan nama tabel yang ingin dibuat setelah kata kunci CREATE TABLE. Nama tabel harus unik dalam database. JIKA TIDAK ADA adalah opsional. Ini memungkinkan  untuk memeriksa apakah tabel yang dibuat sudah ada di database. Jika ini masalahnya, MySQL akan mengabaikan seluruh pernyataan dan tidak akan membuat tabel baru.

**Kedua**, tentukan daftar kolom tabel dibagian column\_list, kolom dipisahkan dengan koma.

**Ketiga**, secara opsional tenntukan mesin penyimpanan untuk tabel dalam klausa ENGINE. Dapat menggunakan mesin penyimpanan apapun seperti InnoDB dan MyISAM. Jika tidak mendeklarasikan mesin penyimpanan secara eksplisit, MySQL akan menggunakan InnoDB secara default.

InnoDB menjadi mesin penyimpanan default sejak MySQL versi 5.5. Mesin penyimpanan InnoDB membawa banyak manfaat dari sistem manajemen basis data relasional seperti transaksi ACID, integritas referensial, dan pemulihan kerusakan. Pada versi sebelumnya, MySQL menggunakan MyISAM sebagai mesin penyimpanan default.

Berikut ini menunjukkan sintaks untuk definisi kolom:

column\_name data\_type(length) [NOT NULL] [DEFAULT value] [AUTO\_INCREMENT] column\_constraint;

Berikut rinciannya:

column\_name menentukan nama kolom. Setiap kolom memiliki tipe data spesifik dan ukuran opsional, mis., VARCHAR(255)

Batasan NOT NULL memastikan bahwa kolom tidak akan berisi NULL. Selain kendala NOT NULL, kolom mungkin memiliki kendala tambahan seperti CHECK, dan UNIQUE.

DEFAULT menentukan nilai default untuk kolom.

AUTO\_INCREMENT menunjukkan bahwa nilai kolom bertambah satu secara otomatis setiap kali baris baru dimasukkan ke dalam tabel. Setiap tabel memiliki maksimum satu kolom AUTO\_INCREMENT.

Setelah daftar kolom,  dapat menentukan batasan tabel seperti UNIQUE, CHECK, PRIMARY KEY dan FOREIGN KEY.

Misalnya, jika ingin menetapkan kolom atau grup kolom sebagai kunci utama, gunakan sintaks berikut:

PRIMARY KEY (col1,col2,...)

## 4.2. MySQL CREATE TABLE statement examples

Beberapa contoh membuat tabel baru.

1) MySQL CREATE TABLE contoh sederhana

Pernyataan berikut membuat tabel baru bernama tasks

CREATE TABLE IF NOT EXISTS tasks (

task\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

start\_date DATE,

due\_date DATE,

status TINYINT NOT NULL,

priority TINYINT NOT NULL,

description TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

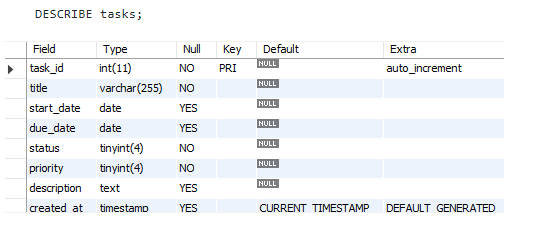
) ENGINE=INNODB;

Tabel tasks memiliki kolom berikut ini:

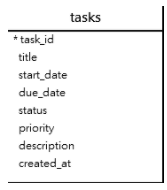
* + Task\_id adalah kolom kenaikan otomatis. Jika  menggunakan pernyataan INSERT untuk menyisipkan baris baru ke dalam tabel tanpa menentukan nilai untuk kolom id\_tugas, MySQL akan secara otomatis menghasilkan bilangan bulat berurutan untuk id\_tugas mulai dari 1.
  + Kolom judul adalah kolom string karakter variabel yang panjang maksimumnya adalah 255. Artinya,  tidak dapat memasukkan string yang panjangnya lebih besar dari 255 ke dalam kolom ini. Batasan NOT NULL menunjukkan bahwa kolom tidak menerima NULL. Dengan kata lain,  harus memberikan nilai non-NULL saat menyisipkan atau memperbarui kolom ini.
  + Start\_date dan due\_date adalah kolom DATE. Karena kolom ini tidak memiliki batasan NOT NULL, kolom ini dapat menyimpan NULL. Kolom start\_date memiliki nilai default dari tanggal saat ini. Dengan kata lain, jika  tidak memberikan nilai untuk kolom tanggal\_mulai saat  menyisipkan baris baru, kolom tanggal\_mulai akan mengambil tanggal saat ini dari server database.
  + Status dan prioritas adalah kolom TINYINT yang tidak mengizinkan NULL.
  + Kolom deskripsi adalah kolom TEXT yang menerima NULL.
  + Created\_at adalah kolom TIMESTAMP yang menerima waktu saat ini sebagai nilai default.
  + task\_id adalah kolom kunci utama dari tabel tugas. Ini berarti bahwa nilai dalam kolom task\_id akan secara unik mengidentifikasi baris dalam tabel.

Setelah menjalankan pernyataan CREATE TABLE untuk membuat tabel tasks,  dapat melihat strukturnya dengan menggunakan pernyataan DESCRIBE:

DESCRIBE tasks;



Gambar ini menunjukkan diagram database dari tabel tasks:



## 4.3. MySQL CREATE TABLE with a foreign key primary key example

Misalkan setiap "task" memiliki daftar periksa atau daftar tugas. Untuk menyimpan daftar periksa tugas, Anda dapat membuat tabel baru bernama "checklists" sebagai berikut:

CREATE TABLE IF NOT EXISTS checklists (

todo\_id INT AUTO\_INCREMENT,

task\_id INT,

todo VARCHAR(255) NOT NULL,

is\_completed BOOLEAN NOT NULL DEFAULT FALSE,

PRIMARY KEY (todo\_id , task\_id),

FOREIGN KEY (task\_id)

REFERENCES tasks (task\_id)

ON UPDATE RESTRICT ON DELETE CASCADE

);

Daftar "checklists" memiliki kunci utama yang terdiri dari dua kolom. Oleh karena itu, kami menggunakan batasan tabel untuk mendefinisikan [primary key](https://www.mysqltutorial.org/mysql-primary-key/):

PRIMARY KEY (todo\_id , task\_id)

Selain itu, task\_id adalah kolom foreign key column yang mengacu pada task\_id column pada table tasks,

menggunakan foreign key constraint untuk relationsgip:

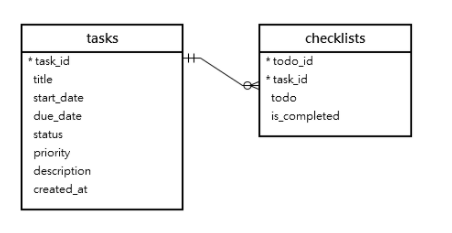
FOREIGN KEY (task\_id)

REFERENCES tasks (task\_id)

ON UPDATE RESTRICT

ON DELETE CASCADE

Gambar ini menggambarkan tabel daftar periksa dan hubungannya dengan tabel tugas:



## 4.4. MySQL Sequence

Di MySQL, urutan adalah daftar dari [integers](https://www.mysqltutorial.org/mysql-int/) dihasilkan dalam urutan ascending yaitu, 1,2,3… Banyak aplikasi membutuhkan urutan untuk menghasilkan nomor unik terutama untuk identifikasi misalnya, ID pelanggan di CRM, nomor karyawan di HR, dan nomor peralatan dalam sistem manajemen layanan.

Untuk membuat urutan di MySQL secara otomatis,  atur: atribut untuk sebuah kolom AUTO\_INCREMENT, yang biasanya adalah kolom [primary key](https://www.mysqltutorial.org/mysql-primary-key/).

Aturan berikut diterapkan saat menggunakan atribut AUTO\_INCREMENT:

* Setiap tabel hanya memiliki 1 kolom AUTO\_INCREMENT dimana tipe datanya adalah [integer](https://www.mysqltutorial.org/mysql-int/).
* Kolom AUTO\_INCREMENT harus di index, yang berarti bisa juga [PRIMARY KEY](https://www.mysqltutorial.org/mysql-primary-key/) atau [UNIQUE](https://www.mysqltutorial.org/mysql-unique/) index.
* Kolom AUTO\_INCREMENT harus bersifat  [NOT NULL constraint](https://www.mysqltutorial.org/mysql-not-null-constraint/). Saat mengatur atribut kolom AUTO\_INCREMENT , MySQL secara otomatis menambhakan NOT NULL  ke kolom secara implisit

**Creating MySQL sequence example**

Berikut perintah  [creates a table](https://www.mysqltutorial.org/mysql-create-table/) dengan nama employees memiliki kolom emp\_no yang  AUTO\_INCREMENT :

CREATE TABLE employees (

emp\_no INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50)

);

**How MySQL sequence works**

Kolom AUTO\_INCREMENT memilki atribut berikut ini:

* Nilai dari awal kolom AUTO\_INCREMENT adalah 1 dan bertambah 1 saat  memasukkan  [NULL](https://www.mysqltutorial.org/mysql-null/)value kedalam kolom atau ketika  menghilangkan nilainya dalam [INSERT](https://www.mysqltutorial.org/mysql-insert-statement.aspx) statement.
* Untuk mendapatkan nomor urut yang dihasilkan terakhir, gunakan [LAST\_INSERT\_ID()](https://www.mysqltutorial.org/mysql-last_insert_id.aspx)  function.
* Menggunakan ID penyisipan terakhir untuk pernyataan berikutnya, misalnya, memasukkan data ke dalam tabel. Urutan yang dihasilkan terakhir bersifat unik di seluruh sesi. Dengan kata lain, jika koneksi lain menghasilkan nomor urut, dapat memperolehnya denga menggunakan LAST\_INSERT\_ID() function.
* JIka [insert a new row into a table](https://www.mysqltutorial.org/mysql-insert-statement.aspx) dan tentukan nilai untuk kolom urut, MySQL akan memasukkan nomor urut jika nomor urut tidak ada dikolom atau mengeluarkan kesalahan jika sudah ada. Jika  memasukkan nilai baru yang lebih besar dari nomor urut berikutnya, MySQL akan menggunakan nilai baru sebagai nomor urut awal dan menghasilkan nomor urut unik yang lebih besar dari yang sekarang untuk penggunaan berikutnya. Ini menciptakan kesenjangan dalam urutan.
* Jika menggunakan [UPDATE](https://www.mysqltutorial.org/mysql-update-data.aspx) untuk merubah nilai kolom AUTO\_INCREMENT  ke nilai yang sudah ada, MySQL akan mengeluarkan kesalahan kunci duplikat jika kolom tersebut memiliki indeks unik. Jika  memperbarui kolom AUTO\_INCREMENT kolom ke nilai yang lebih besar dari nilai yang ada dikolom, MySQL akan menggunakan nomor berikutnya dari nomor urut sisipkan terakhir untuk baris berikutnya. Misalnya, jika nomor urut sisipan terakhir adalah 3,  memperbaruinya menjadi 10, nomor urut untuk baris baru adalah 4
* Jika menggunakan [DELETE](https://www.mysqltutorial.org/mysql-delete-statement.aspx) statement untuk menghapus baris yang terakhir disisipkan, MySQL  tidak menggunakan nomor urut yang dihapus tergantung pada tabel [storage engine](https://www.mysqltutorial.org/understand-mysql-table-types-innodb-myisam.aspx) . Tabel MyISAM tidak menggunakan kembali nomor urut yang dihapus jika Anda menghapus baris misalnya, id sisipan terakhir dalam tabel adalah 10, jika Anda menghapusnya, MySQL masih menghasilkan nomor urut berikutnya yaitu 11 untuk baris baru. Mirip dengan tabel MyISAM, tabel InnoDB tidak menggunakan kembali nomor urut saat baris dihapus.

Setelah mengatur atribut AUTO\_INCREMENT ke kolom, dapat [reset the auto increment](https://www.mysqltutorial.org/mysql-reset-auto-increment) nilai dalam berbagai cara misalnya, dengan menggunakan [ALTER TABLE](https://www.mysqltutorial.org/mysql-alter-table.aspx) statement.

Beberapa contoh untuk mendapatkan pemahaman yang lebih baik tentang urutan MySQL.

Pertama, insert 2 baris baru ke tabel employees :

INSERT INTO employees(first\_name,last\_name)

VALUES('John','Doe'),

('Mary','Jane');

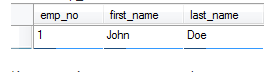
Kedua, select data dari tabel employees :

SELECT \* FROM employees;

Ketiga, hapus employee kedua dengan emp\_no = 2:

DELETE FROM employees

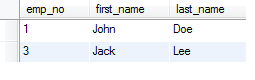
WHERE emp\_no = 2;



Keempat, insert a new employee:

INSERT INTO employees(first\_name,last\_name)

VALUES('Jack','Lee');



karena storage engine pada tabel employees adalah InnoDB, tidak menggunakan kembali nomor urut yang dihapus. Baris baru memiliki  emp\_no  3.

Kelima, memperbarui karyawan yang ada dengan  emp\_no 3 ke 1:

UPDATE employees

SET

first\_name = 'Joe',

emp\_no = 1

WHERE

emp\_no = 3;

Terdapat kesalahan entri duplikat untuk primary key. Perbaikan:

UPDATE employees

SET

first\_name = 'Joe',

emp\_no = 10

WHERE

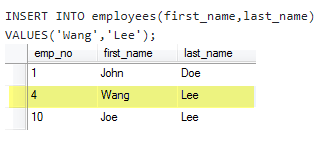
emp\_no = 3;



Keenam, masukkan karyawan baru setelah memperbarui nomor urut menjadi 10:

INSERT INTO employees(first\_name,last\_name)

VALUES('Wang','Lee');



Nomor urut berikutnya dari sisipan terakhir adalah nomor 4, oleh karena itu, MySQL menggunakan nomor 4 untuk baris baru, bukan 11.

## 4.5. Create Table Using Another Table

Salinan tabel yang ada juga dapat dibuat menggunakan CREATE TABLE.

Tabel baru mendapatkan definisi kolom yang sama. Semua kolom atau kolom tertentu dapat dipilih.  
Jika  membuat tabel baru menggunakan tabel yang sudah ada, tabel baru akan diisi dengan nilai yang ada dari tabel lama.

**Syntax**

CREATE TABLE new\_table\_name AS  
    SELECT column1, column2,...    FROM existing\_table\_name    WHERE ....;

SQL berikut membuat tabel baru yang disebut "TestTables" (yang merupakan salinan dari tabel "customers"):

Contoh:

CREATE TABLE TestTable AS  
SELECT customername, contactname  
FROM customers;

Latihan:

Tulis pernyataan SQL yang benar untuk membuat tabel baru bernama Persons.

....................... (

PersonID int,

LastName varchar(255),

FirstName varchar(255),

Address varchar(255),

City varchar(255)

);

## 5.1. RENAME TABLE statement

Karena persyaratan bisnis berubah, kita perlu mengganti nama tabel saat ini menjadi tabel baru untuk lebih mencerminkan situasi baru. MySQL memberi  pernyataan yang sangat berguna yang mengubah nama satu atau lebih tabel.  
Untuk mengubah satu atau lebih tabel,  menggunakan  RENAME TABLE statement berikut :

RENAME TABLE old\_table\_name TO new\_table\_name;

Tabel lama ( old\_table\_name) harus ada, dan tabel baru ( new\_table\_name) tidak harus ada. Jika tabel baru new\_table\_name tidak ada, pernyataan akan gagal

Selain tabel,  dapat menggunakan RENAME TABLE statement ke rename [view](https://www.mysqltutorial.org/mysql-views-tutorial.aspx)s.

Sebelum mengeksekusi  RENAME TABLE statement, harus memastikan tidak ada transaksi aktif atau [locked tables](https://www.mysqltutorial.org/mysql-table-locking/).

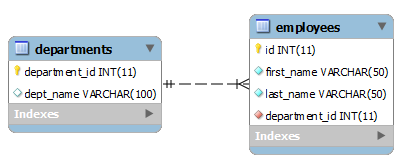
Dalam hal keamanan, apapun yang ada [privileges that we granted to the old table](https://www.mysqltutorial.org/mysql-grant.aspx) harus dimigrasikan secara manual ke tabel baru.

Sebelum mengganti nama tabel, Anda harus mengevaluasi dampaknya secara menyeluruh. Misalnya, Anda harus menyelidiki aplikasi mana yang menggunakan tabel. Jika nama tabel berubah, maka kode aplikasi yang mengacu pada nama tabel tersebut juga perlu diubah. Selain itu, Anda harus menyesuaikan objek database lain secara manual seperti [views](https://www.mysqltutorial.org/mysql-views-tutorial.aspx), [stored procedures](https://www.mysqltutorial.org/mysql-stored-procedure-tutorial.aspx), [triggers](https://www.mysqltutorial.org/mysql-triggers.aspx), [foreign key constraints](https://www.mysqltutorial.org/mysql-foreign-key/), dll.,

referensi ke tabel.

**Contoh**

Pertama,  [create a new database](https://www.mysqltutorial.org/mysql-create-drop-database.aspx) dengan nama hr terdiri dari 2 tabel employees dan departments untuk demonstrasi.



CREATE DATABASE IF NOT EXISTS hr;

CREATE TABLE departments (

department\_id INT AUTO\_INCREMENT PRIMARY KEY,

dept\_name VARCHAR(100)

);

CREATE TABLE employees (

id int AUTO\_INCREMENT primary key,

first\_name varchar(50) not null,

last\_name varchar(50) not null,

department\_id int not null,

FOREIGN KEY (department\_id)

REFERENCES departments (department\_id)

);

Kedua,  [insert sample data](https://www.mysqltutorial.org/mysql-insert-statement.aspx) kedalam 2 tabel employees dan departments:

INSERT INTO departments(dept\_name)

VALUES('Sales'),('Markting'),('Finance'),('Accounting'),('Warehouses'),('Production');

INSERT INTO employees(first\_name,last\_name,department\_id)

VALUES('John','Doe',1),

('Bush','Lily',2),

('David','Dave',3),

('Mary','Jane',4),

('Jonatha','Josh',5),

('Mateo','More',1);

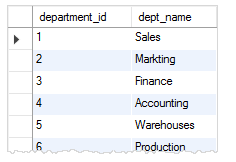
Ketiga, tinjau data pada tabel departments dan employees:

SELECT

department\_id, dept\_name

FROM

departments;

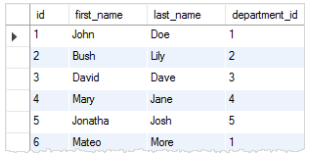


SELECT

id, first\_name, last\_name, department\_id

FROM

employees;



## 5.1.1. Mengganti nama tabel yang direferensikan oleh tampilan

Jika tabel yang akan Anda ganti namanya direferensikan oleh [view](https://www.mysqltutorial.org/mysql-views-tutorial.aspx), tampilan akan menjadi tidak valid jika Anda mengganti nama tabel, dan Anda harus menyesuaikan tampilan secara manual.

Misalnya,  membuat tampilan bernama v\_employee\_info berdasarkan tabel employees dan departments sebagai berikut:

CREATE VIEW v\_employee\_info as

SELECT

id, first\_name, last\_name, dept\_name

from

employees

inner join

departments USING (department\_id);

Tampilan menggunakan [inner join](https://www.mysqltutorial.org/mysql-inner-join.aspx) klausa untuk menggabung tabel  departments dan employees.

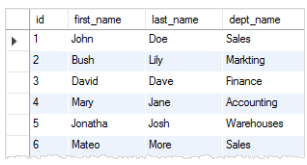
Berikut [SELECT statement](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) mengembalikan semua data dari tanpilan v\_employee\_info.

SELECT

\*

FROM

v\_employee\_info;



Ganti nama tabel employees menjadi people dan query data dari tampilan v\_employee\_info.

RENAME TABLE employees TO people;

SELECT

\*

FROM

v\_employee\_info;

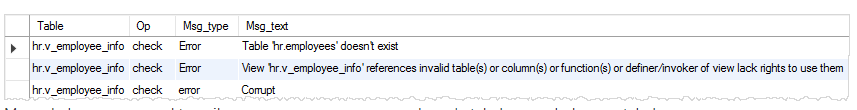
**Error message:**

Error Code: 1356. View 'hr.v\_employee\_info' references invalid table(s) or

column(s) or function(s) or definer/invoker of view lack rights to use them

Menggunakan CHECK TABLE statement untuk memeriksa tampilan status v\_employee\_info berikut:

CHECK TABLE v\_employee\_info;



Mengubah secara manual tampilan v\_employee\_info mengacu kepada tabel people bukannya tabel employees .

Jika tabel yang diganti namanya dirujuk oleh  [stored procedure](https://www.mysqltutorial.org/mysql-stored-procedure-tutorial.aspx), menyesuaikannya secara manual seperti yang  dilakukan dengan tampilan:

Pertama, rename tabel people kembali menjadi tabel employees .

RENAME TABLE people TO employees;

Lalu, [create a new stored procedure](https://www.mysqltutorial.org/getting-started-with-mysql-stored-procedures.aspx) dengan nama get\_employee yang mengacu pada tabel employees.

DELIMITER $$

CREATE PROCEDURE get\_employee(IN p\_id INT)

BEGIN

SELECT first\_name

,last\_name

,dept\_name

FROM employees

INNER JOIN departments using (department\_id)

WHERE id = p\_id;

END $$

DELIMITER;

Lalu, eksekusi tabel get\_employee untuk mendapatkan data karyawan dengan id 1 sebagai berikut:

CALL get\_employee(1);



setelah itu, rename tabel employees ke tabel  people.

RENAME TABLE employees TO people;

Akhirnya, disebut dengan get\_employee stored procedure untuk mendapatkan informasi karyawan dengan id 2:

CALL get\_employee(2);

MySQL returns the following error message:

Error Code: 1146. Table 'hr.employees' doesn't exist

Untuk memperbaiki, dirubah secara manual employees table kedalam stored procedure ke tabel people.

## 5.1.2. Mengganti nama tabel yang memiliki kunci asing yang dirujuk

Tabel departments berhubungan dengan tabel employees menggunakan kolom department\_id . Kolom department\_id ditabel employees adalah [foreign key](https://www.mysqltutorial.org/mysql-foreign-key/)  referensi dari tabel departments.

JIka mengganti nama tabel departments, semua foreign keys pada tabel departments tidak akan  diperbarui secara otomatis. Dalam kasus seperti itu,

harus menghapus dan membuat ulang kunci asing secara manual.

RENAME TABLE departments TO depts;

Menghapus departemen dengan id 1, karena batasan foreign key berada pada tabel people yang akan dihapus juga.

Namun, mengganti tabel departments ke tabel depts.

tanpa mengupdate  foreign key dengan manual, MySQL menampilkan pesan error berikut ini:

DELETE FROM depts

WHERE

department\_id = 1;

Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`hr`.`people`, CONSTRAINT `people\_ibfk\_1` FOREIGN KEY (`department\_id`) REFERENCES `depts` (`department\_id`))

**Renaming multiple tables**

Menggunakan RENAME TABLE statement ke rename multiple tables pada waktu yang sama. Dengan statement:

RENAME TABLE old\_table\_name\_1 TO new\_table\_name\_2,

old\_table\_name\_2 TO new\_table\_name\_2,...

Berikut perintah mengganti tabel  people dan depts ke tabel employees dan departments:

RENAME TABLE depts TO departments,

people TO employees;

 RENAME TABLE statement tidak atomik. jika terjadi kesalahan, MySQL melakukan rollback semua tabel yang diubah namanya ke nama lamanya.

**Renaming tables using ALTER TABLE statement**

Merubah tabel dengan ALTER TABLE statement berikut ini:

ALTER TABLE old\_table\_name

RENAME TO new\_table\_name;

 ALTER TABLE statement dapat mengganti nama tabel,  sementara  RENAME TABLE statement tidak dapat.

**Renaming temporary table example**

Pertama, membuat [temporary table](https://www.mysqltutorial.org/mysql-temporary-table/) yang berisi semua nama belakang unik yang berasal dari kolom last\_name pada tabel employees:

CREATE TEMPORARY TABLE lastnames

SELECT DISTINCT last\_name from employees;

Kedua, gunakan  RENAME TABLE untuk merename tabel lastnames:

RENAME TABLE lastnames TO unique\_lastnames;

 Rrror message:

Error Code: 1017. Can't find file: '.\hr\lastnames.frm' (errno: 2 - No such file or directory)

Ketiga, gunakan ALTER TABLE statement untuk merename tabel lastnames.

ALTER TABLE lastnames

RENAME TO unique\_lastnames;

Keempat, query data dari tabel  unique\_lastnames sementara:

SELECT

last\_name

FROM

unique\_lastnames;



## 5.2.1. ALTER TABLE – Modify columns

**1) Modify a column**

Berikut adalah sintaks dasar untuk memodifikasi kolom dalam tabel:

ALTER TABLE table\_name

MODIFY column\_name column\_definition

[ FIRST | AFTER column\_name];

Untuk melihat atribut kolom sebelum memodifikasinya.

Misalkan ingin merubah kolom note  [NOT NULL](https://www.mysqltutorial.org/mysql-not-null-constraint/) dengan maximum 100 characters.

Pertama, tampilkan daftar kolom dari vehicles:

DESCRIBE vehicles;

Lalu, modifikasi kolom note:

ALTER TABLE vehicles

MODIFY note VARCHAR(100) NOT NULL;

Finally, tampilkan daftar kolom dari tabel vehicles untuk memverifikasi perubahan:

DESCRIBE vehicles;

**2) Modify multiple columns**

Pernyataan berikut  untuk mengubah beberapa kolom:

ALTER TABLE table\_name

MODIFY column\_name column\_definition

[ FIRST | AFTER column\_name],

MODIFY column\_name column\_definition

[ FIRST | AFTER column\_name],

...;

Pertama, tunjukkan kolom saat ini dari vehicles table:

Kedua, gunakan ALTER TABLE MODIFY statement untuk memodifikasi beberapa kolom:

ALTER TABLE vehicles

MODIFY year SMALLINT NOT NULL,

MODIFY color VARCHAR(20) NULL AFTER make;

Contoh:

* Pertama, ubah tipe data  kolom year dari INT ke SMALLINT
* Kedua, modifikasi kolom color dengan mengatur panjang maximum sampai 20, hapus NOT NULL constraint, dan mengubah posisinya untuk muncul setelah  
  make column.

Ketiga, tampilkan daftar kolom baru dari vehicles table untuk verifikasi modifikasi.

**MySQL ALTER TABLE – Rename a column in a** table

Untuk mengganti nama kolom, gunakan pernyataan berikut:

ALTER TABLE table\_name

CHANGE COLUMN original\_name new\_name column\_definition

[FIRST | AFTER column\_name];

Dalam sintaks ini:

* Pertama, tentukan nama tabel tempat kolom tersebut berada.
* Kedua, tentukan nama kolom dan nama baru diikuti dengan definisi kolom setelah CHANGE COLUMN keywords.
* Ketiga, gunakan FIRST atau AFTER column\_name pilihan untuk menentukan posisi kolom baru.

Contoh berikut menggunakan ALTER TABLE CHANGE COLUMN statement untuk mengganti kolom  note ke vehicleCondition:

ALTER TABLE vehicles

CHANGE COLUMN note vehicleCondition VARCHAR(100) NOT NULL;

Tinjau daftar kolom dari vehicles table:

DESCRIBE vehicles;

**MySQL ALTER TABLE – Drop a column**

Untuk [drop a column](https://www.mysqltutorial.org/mysql-drop-column/) dalam tabel, gunakan ALTER TABLE DROP COLUMN statement:

ALTER TABLE table\_name

DROP COLUMN column\_name;

Dalam sintaks ini:

* Pertama, tentukan nama tabel yang ingin Anda letakkan kolomnya setelah ALTER TABLE keywords.
* Kedua, tentukan nama kolom yang ingin Anda jatuhkan setelah DROP COLUMN keywords.

Contoh ini menunjukkan bagaimana menghapus kolom vehicleCondition dari tabel  vehicles:

ALTER TABLE vehicles

DROP COLUMN vehicleCondition;

**MySQL ALTER TABLE – Rename table**

Untuk [rename a table](https://www.mysqltutorial.org/mysql-rename-table/), gunakan ALTER TABLE RENAME TO statement:

ALTER TABLE table\_name

RENAME TO new\_table\_name;

Dalam SIntaks ini

* + Pertama, tentukan nama tabel yang ingin Anda ganti namanya setelah ALTER TABLE keywords.
  + Kedua, tentukan nama baru untuk tabel setelah RENAME TO keywords.

Contoh ini mengganti nama tabel vehicles menjadi cars:

ALTER TABLE vehicles

RENAME TO cars;

## 5.3. Drop Tabel

Untuk menghapus tabel yang ada gunakan DROP TABLE statement.

Berikut adalah sintaks dasar DROP TABLE statement:

DROP [TEMPORARY] TABLE [IF EXISTS] table\_name [, table\_name] ...

[RESTRICT | CASCADE]

 DROP TABLE statement menghapus tabel dan datanya secara permanen dari database. Di MySQL,  juga dapat menghapus beberapa tabel menggunakan satu DROP TABLE statement, setiap tabel dipisahkan dengan koma(,).

 TEMPORARY opsi memungkinkan  untuk menghapus hanya [temporary tables](https://www.mysqltutorial.org/mysql-temporary-table/) . Ini memastikan bahwa   secara tidak sengaja menghapus tabel non-sementara.

IF EXISTS adalah opsi menghapus tabel secara kondisional hanya jika ada. Jika menghapus tabel yang tidak ada dengan IF EXISTS option, MySQL menghasilkan CATATAN, yang dapat diambil menggunakan  
[SHOW WARNINGS](https://www.mysqltutorial.org/mysql-stored-procedure/mysql-show-warnings/) statement.

Perhatikan bahwa DROP TABLE statement hanya menghapus tabel. Tidak menghapus hak pengguna tertentu yang terkait dengan tabel. Oleh karena itu, jika  membuat tabel dengan nama yang sama dengan yang dihapus, MySQL akan menerapkan hak istimewa yang ada ke tabel baru, yang dapat menimbulkan risiko keamanan.

 RESTRICT dan CASCADE  opsi dicadangkan untuk versi MySQL yang akan datang.

Untuk mengeksekusi DROP TABLE statement, DROP adalah hak istimewa untuk tabel yang ingin  dihapus.

**Contoh**

Beberapa contoh menggunakan DROP TABLE statement.

**A) Using MySQL DROP TABLE to drop a single table example**

Pertama, buat nama tabel insurances untuk tujuan pengujian:

CREATE TABLE insurances (

id INT AUTO\_INCREMENT,

title VARCHAR(100) NOT NULL,

effectiveDate DATE NOT NULL,

duration INT NOT NULL,

amount DEC(10 , 2 ) NOT NULL,

PRIMARY KEY(id)

);

Kedua, gunakan DROP TABLE untuk menghapus insurances table:

DROP TABLE insurances;

**A) Using MySQL DROP TABLE to drop multiple tables**

First, create two tables named CarAccessories and CarGadgets:

CREATE TABLE CarAccessories (

id INT AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

price DEC(10 , 2 ) NOT NULL,

PRIMARY KEY(id)

);

CREATE TABLE CarGadgets (

id INT AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

price DEC(10 , 2 ) NOT NULL,

PRIMARY KEY(id)

);

Kedua, gunakan DROP TABLE statement untuk menghapus 2 tabel:

DROP TABLE CarAccessories, CarGadgets;

**C) Using MySQL DROP TABLE to drop a non-existing table**

Pernyataan ini mencoba untuk menjatuhkan tabel yang tidak ada:

DROP TABLE aliens;

MySQL issued the following error:

Error Code: 1051. Unknown table 'classicmodels.aliens'

Namun, jika menggunakan IF EXISTS pilihan DROP TABLE statement:

DROP TABLE IF EXISTS aliens;

MySQL mengeluarkan peringatan sebagai gantinya:

0 row(s) affected, 1 warning(s): 1051 Unknown table 'classicmodels.aliens'

Untuk menampilkan peringatan,  dapat menggunakanSHOW WARNINGS statement:

SHOW WARNINGS;

## Rangkuman Topik 5

Alter adalah menrubah struktur tabel, seperti : menambah kolom, menghapus kolom atau mengganti kolom.  Renama adalah merubah nama tabel dan bila tabel yang di rename maka tabel yang berhubungan dengan tabel tersebut akan berubah namanya. Drop adalah menghapus tabel, dan juga tabel yang berhubungan dengan tabel tersebut akan terhapus, ini terjadi bila tabel tersebut tidak digunakan kembali atau dapat di sederhanakan. Apabila tabel tersebut dihapus, maka isi tabel juga ikut terhapus semua.

## 6.1. Primary Key

Primary Key/Kunci utama adalah kolom atau kumpulan kolom yang secara unik mengidentifikasi setiap baris dalam tabel. Kunci utama mengikuti aturan ini:

* Primary key harus berisi nilai unik. Jika kunci utama terdiri dari beberapa kolom, kombinasi nilai dalam kolom ini harus unik.
* Primary tidak boleh [NULL](https://www.mysqltutorial.org/mysql-null/) values. Setiap [insert](https://www.mysqltutorial.org/mysql-insert-statement.aspx) atau [update](https://www.mysqltutorial.org/mysql-update-data.aspx) NULL  primary key akan error. Perhatikan bahwa MySQL secara implisit menambahkan  NOT NULLconstraint to primary key columns.
* Sebuah tabel dapat memiliki satu primary key saja.

Karena MySQL lebih cepat  daripada integers,  [data type](https://www.mysqltutorial.org/mysql-data-types.aspx)  primary key harus  integer, [INT](https://www.mysqltutorial.org/mysql-int/), BIGINT. Pastikan bahwa rentang nilai tipe integer untuk kunci utama cukup untuk menyimpan semua kemungkinan baris yang mungkin dimiliki tabel.

Primary key sering memilki  [AUTO\_INCREMENT](https://www.mysqltutorial.org/mysql-sequence/) attribute yang secara otomatis menghasilkan bilangan bulat berurutan setiap kali  [insert a new row](https://www.mysqltutorial.org/mysql-insert-statement.aspx) kedalam table.

Saat Anda menentukan kunci utama untuk sebuah tabel, MySQL secara otomatis [creates an index](https://www.mysqltutorial.org/mysql-index/mysql-create-index/) disebut dengan  PRIMARY.

## 6.1.2. Contoh PK

PRIMARY KEY  pada  [create](https://www.mysqltutorial.org/mysql-create-table/) or alter table.

1)  PRIMARY KEY pada CREATE TABLE and alter table

Menentukan primary key pada [CREATE TABLE](https://www.mysqltutorial.org/mysql-create-table/).

1 PRIMARY KEY  pada sebuah tabel:

CREATE TABLE table\_name(

primary\_key\_column datatype PRIMARY KEY,

...

);

 1 atau lebih PRIMARY KEY pada sebuah tabel.

CREATE TABLE table\_name(

primary\_key\_column1 datatype,

primary\_key\_column2 datatype,

...,

PRIMARY KEY(column\_list)

);

Memisahkan Kolom column\_list dengan commas (,).

 PRIMARY KEY pada batasan tabel dapat digunakan ketika primary key memiliki satu kolom:

CREATE TABLE table\_name (

primary\_key\_column datatype,

... ,

PRIMARY KEY(primary\_key\_column)

);

Contoh [creates a table](https://www.mysqltutorial.org/mysql-create-table/) dengan Nama Tabel  users dengan Nama Kolom primary key user\_id :

CREATE TABLE users(

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(40),

password VARCHAR(255),

email VARCHAR(255)

);

Perintah membuat Tabel roles dengan PRIMARY KEY sebagai berikut:

CREATE TABLE roles(

role\_id INT AUTO\_INCREMENT,

role\_name VARCHAR(50),

PRIMARY KEY(role\_id)

);

Jika kunci utama terdiri dari beberapa kolom, tentukan diakhir perintah  CREATE TABLE .  Daftar primary key yang dipisahkan koma didalam tanda kurung diikuti dengan PRIMARY KEY .

Contoh membuat Tabel  user\_roles  yang primary key Nya terdiri dari dua kolom user\_id dan role\_id.  PRIMARY KEY didefenisikan dengan:

CREATE TABLE user\_roles(

user\_id INT,

role\_id INT,

PRIMARY KEY(user\_id,role\_id),

FOREIGN KEY(user\_id)

REFERENCES users(user\_id),

FOREIGN KEY(role\_id)

REFERENCES roles(role\_id)

);

Perhatikan bahwa pernyataan itu juga menciptakan dua  [foreign key](https://www.mysqltutorial.org/mysql-foreign-key/)

2)  PRIMARY KEY menggunaan ALTER TABLE

Jika sebuah tabel, karena alasan tertentu, tidak memiliki primary key,  dapat menggunakan

Perintah  [ALTER TABLE](https://www.mysqltutorial.org/mysql-alter-table.aspx) untuk menambahkan  primary key the tabel:

ALTER TABLE table\_name

ADD PRIMARY KEY(column\_list);

Contoh menambahkan Koloa  id pada primary key.

Pertama.  Buatan Tabel  pkdemos tanpa primary key.

CREATE TABLE pkdemos(

id INT,

title VARCHAR(255) NOT NULL

);

Kedua, tambahkan  pkdemos pada tabel menggunakan peritnah ALTER TABLE :

ALTER TABLE pkdemos

ADD PRIMARY KEY(id);

Jika Anda menambahkan kunci utama ke tabel yang sudah memiliki data. Data dalam kolom yang akan dimasukkan ke dalam kunci utama harus unik dan bukan NULL.

**PRIMARY KEY vs. UNIQUE KEY vs. KEY**

KEY ada synonym untuk INDEX. Gunakan  KEY Ketika ingin membuat [create an index](https://www.mysqltutorial.org/mysql-index/mysql-create-index/) untuk kolom atau kumpulan kolom yang bukan merupakan bagian dari kunci utama atau [unique key](https://www.mysqltutorial.org/mysql-unique/).

Ineks [UNIQUE](https://www.mysqltutorial.org/mysql-unique/) index memastikan bahwa nilai dalam kolom harus unik. tidak seperti index PRIMARY , memugkinkan nilai  NULL pada index  UNIQUE. Selain itu, sebuah tabel dpat memiliki beberapa index  UNIQUE.

Misal, email dan username of users in the users table must be unique. To enforce thes rules, you can define UNIQUE indexes for the email and username columns as the following  statement:

Tambahkan index  UNIQUE untuk kolom username :

ALTER TABLE users

ADD UNIQUE INDEX username\_unique (username ASC) ;

Tambahan index UNIQUE untuk kolom  email :

ALTER TABLE users

ADD UNIQUE INDEX  email\_unique (email ASC) ;

## 6.1.3. Latihan Praktek pada PK

**Buat  PRIMARY KEY dengan Kolom "ID" dan Nama Tabel "Persons" :**

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    PRIMARY KEY (ID)  
);

**PRIMARY KEY pada beberapa kolom, menggunakan SQL syntax:**

CREATE TABLE Persons (

    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    CONSTRAINT PK\_Person PRIMARY KEY (ID,LastName)  
);

**Membuat PRIMARY KEY dengan Nama Kolom "ID" pada Tabel yang Sudah ada, menggunakan perintah:**

ALTER TABLE Persons  
ADD PRIMARY KEY (ID);

**PRIMARY KEY pada banyak kolom:**

ALTER TABLE Persons  
ADD CONSTRAINT PK\_Person PRIMARY KEY (ID,LastName);

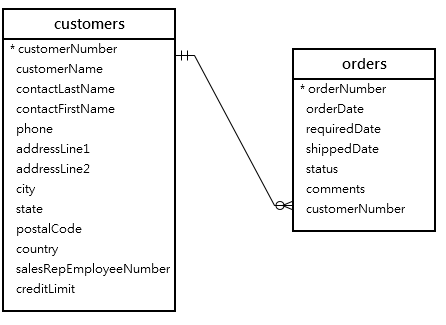
Menghapus PRIMARY KEY menggunakan sintaks:

ALTER TABLE Persons  
DROP PRIMARY KEY;

## 6.2. Foreign Key/FK

Foreign key adalah kolom atau group pada column dalam table yang terhubung dengan kolom atau grub pada kolom di Tabel lainnya. Foreign key menempatkan batasan pada data dalam tabel terkait, yang memungkinkan MySQL untuk mempertahankan integritas referensial.

Berikut Tabel customers dan orders dari  [sample database](https://www.mysqltutorial.org/mysql-sample-database.aspx).



Dalam diagram ini, setiap pelanggan dapat memiliki nol atau banyak pesanan dan setiap pesanan dimiliki oleh satu pelanggan.

Hubungan antara Tabel  customers dan orders adalah one-to-many. Hubungannya pada foreign key di Tabel  orders pada kolom customerNumber.

CustomerNumber di Tabel orders berhubungan dengan primary ke customerNumber pada Tabel customers.

Tabel customers disebut  parent table atau referenced table(Tabel Utama), dan Tabel orders  adalah Tabel transaksi atau Tabel reference.

Secara umum, foreign key pada table transaksi adalah [primary key](https://www.mysqltutorial.org/mysql-primary-key/) pada tabel induk/tabel master.

Satu Tabel dapat memiliki lebih dari 1  foreign key Dimana foreign key lainnya berasalah dari Tabel master/tabel induk/Tabel Utama.

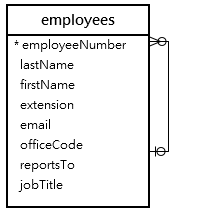
Foreign key dari tabel anak harus memiliki baris yang sesuai dengan kolom kunci induk dari tabel induk atau nilai dalam kolom kunci asing ini harus  NULL (SET NULL vonoth berikut ini).

Contoh, Setiap baris Tabel  orders memiliki satu  kolom customerNumber pada tabel  customers. Banyak baris pada tabel  orders memiliki nama yang sama pada kolom customerNumber.

**Self-referencing Foreign Key/FK**

Terkadang, tabel transaksi dan master dapat merujuk ke tabel yang sama. Dalam hal ini, FK merujuk kembali ke kunci utama dalam tabel yang sama.

Lihat Tabel  employees dari [sample database](https://www.mysqltutorial.org/mysql-sample-database.aspx).



Kolom reportTo adalah  foreign key berasal dari kolom employeeNumber sebagai  primary key pada Tabel employees.

Hubungan tersebut memungkinkan Tabel employees untuk menyimpan struktur pelaporan antara karyawan dan manajer. Setiap karyawan melapor ke nol atau satu karyawan dan seorang karyawan dapat memiliki nol atau banyak bawahan.

Foreign key  reportTo dikenal sebagai rekursif atau referensi diri dari foreign key.

## syntax

Foreign key pada [CREATE TABLE](https://www.mysqltutorial.org/mysql-create-table/) atau [ALTER TABLE](https://www.mysqltutorial.org/mysql-alter-table.aspx) dengan statement:

[CONSTRAINT constraint\_name]

FOREIGN KEY [foreign\_key\_name] (column\_name, ...)

REFERENCES parent\_table(colunm\_name,...)

[ON DELETE reference\_option]

[ON UPDATE reference\_option]

Dalam syntax ini:

Pertama, penamaan  foreign key spesifik yang akan create setelah keyword  CONSTRAINT. Jika menghilangkannya, sacara otomatis  menghasilkan penamaan untuk  foreign key.

Kedua, tentukan daftar yang dipisahkan oleh tanda koma pada foreign key columns setelah keywords FOREIGN KEY. Penamaan foreign key  adalah pilihan  dan dihasilkan secara otomatis jika terabaikan.

Ketiga, tentukan tabel induk diikuti dengan daftar kolom yang dipisahkan koma yang menjadi referensi kolom FK.

Terakhir, tentukan bagaimana FK mempertahankan integritas referensial antara tabel transaksi dan master dengan Klaus ON DELETE dan ON UPDATE.   reference\_option menentukan tindakan yang akan diambil MySQL ketika nilai di kolom PK dihapus (ON DELETE) atau update (ON UPDATE).

MySQL memiliki lima opsi referensi: CASCADE, SET NULL, NO ACTION, RESTRICT, dan SET DEFAULT.

* CASCADE: jika baris dari tabel induk dihapus atau diperbarui, nilai baris yang cocok di tabel transaski/anak secara otomatis dihapus atau diperbarui
* SET NULL: jika baris dari tabel induk dihapus atau diperbarui, nilai kolom FK (atau kolom) di tabel transaksi ditetapkan NULL.
* RESTRICT:  JIka baris dari Tabel induk memiliki baris yang cocok di tabel anak, MySQL menolak menghapus atau memperbarui baris di tabel induk.
* NO ACTION: sama sebagai RESTRICT.
* SET DEFAULT: dikenali oleh parser MySQL. Namun, tindakan ini ditolak oleh tabel InnoDB dan NDB.

Faktanya, MySQL sepenuhnya mendukung tiga tindakan: RESTRICT, CASCADE dan SET NULL.

Jika tidak menentukan Klaus ON DELETE dan ON UPDATE, tindakan default adalah RESTRICT

## Rangkuman Topik 6

**FOREIGN KEY on CREATE TABLE**

The following SQL creates a FOREIGN KEY on the "PersonID" column when the "Orders" table is created:

CREATE TABLE Orders (  
    OrderID int NOT NULL,  
    OrderNumber int NOT NULL,  
    PersonID int,  
    PRIMARY KEY (OrderID),  
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)  
);

To allow naming of a FOREIGN KEY constraint, and for defining a FOREIGN KEY constraint on multiple columns, use the following SQL syntax:

CREATE TABLE Orders (  
    OrderID int NOT NULL,  
    OrderNumber int NOT NULL,  
    PersonID int,  
    PRIMARY KEY (OrderID),  
    CONSTRAINT FK\_PersonOrder FOREIGN KEY (PersonID)  
    REFERENCES Persons(PersonID)  
);

**FOREIGN KEY on ALTER TABLE**

To create a FOREIGN KEY constraint on the "PersonID" column when the "Orders" table is already created, use the following SQL:

ALTER TABLE Orders  
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);

To allow naming of a FOREIGN KEY constraint, and for defining a FOREIGN KEY constraint on multiple columns, use the following SQL syntax:

ALTER TABLE Orders  
ADD CONSTRAINT FK\_PersonOrder  
FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);

DROP a FOREIGN KEY Constraint

To drop a FOREIGN KEY constraint, use the following SQL:

ALTER TABLE Orders  
DROP FOREIGN KEY FK\_PersonOrder;

## 7.1 Program Transaksi Peminjaman Buku

Code pada form Peminjaman Buku

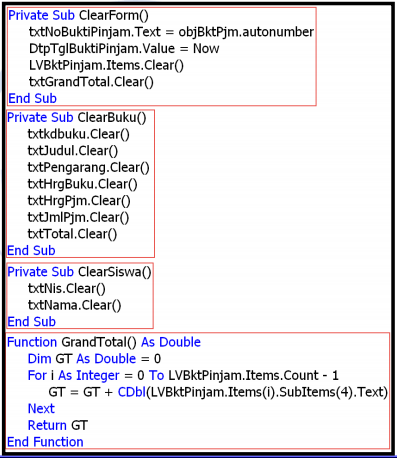
* Buka View Code FrmPinjamBuku pada Solution Explorer



* Kemudian Ketikkan program berikut di bawah Public Class FrmPinjamBuku (Kotak Merah)



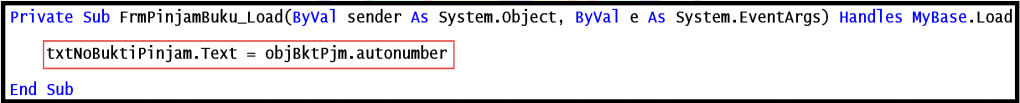
* Ketikkan program prosedur ClearForm, ClearBuku, ClearSiswa, dan Function GrandTotal berurutan setelah perintah Dim objAda As New ClsAda (Kotak Merah)



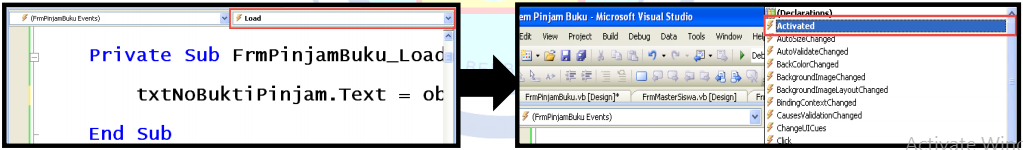
* Klik dua Kali Form pada Desain FrmPinjamBuku



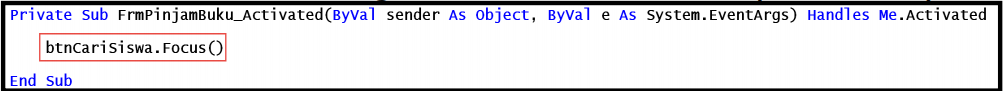
* Kemudian Ketikkan program berikut (Kotak Merah)



* Ubah Event Load Menjadi Activate



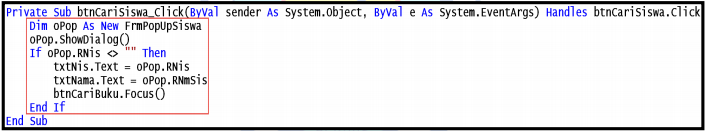
* Kemudian Ketikkan Program Berikut Pada Activated (Kotak Merah)



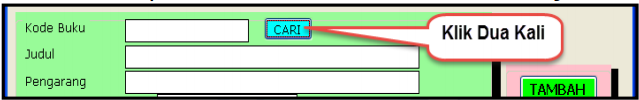
* Untuk tombol cari siswa dan cari buku, akan menampilkan form popup yang sama dengan popup master.
* Maka tidak perlu dibuat lagi
* Gunakan form popup yang sama dengan popup yang ada di master buku dan master siswa.
* Hanya modifikasi code program, disesuaikan dengan field-field yang ditampilkan
* Klik dua Kali pada Tombol Cari dari Desain FrmPinjamBuku



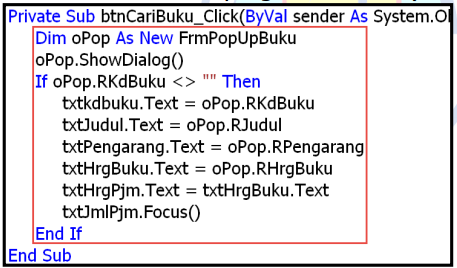
* Kemudian ketikkan program berikut (Kotak Merah)



* Klik dua Kali pada Tombol Cari dari Desain FrmPinjamBuku



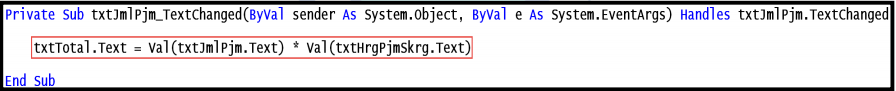
* Kemudian ketikkan program berikut (Kotak Merah)



* Klik dua Kali TextBox Jumlah Pinjam pada Desain FrmPinjamBuku



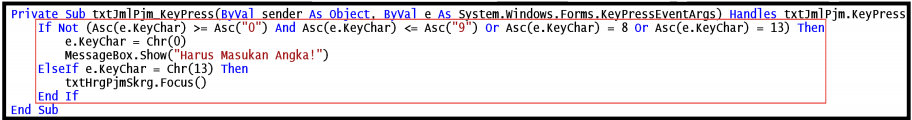
* Kemudian Ketikkan program berikut (Kotak Merah)



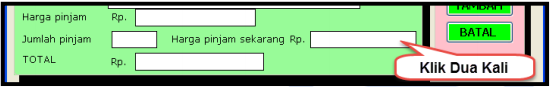
* Ubah Event TextChange Menjadi Keypress



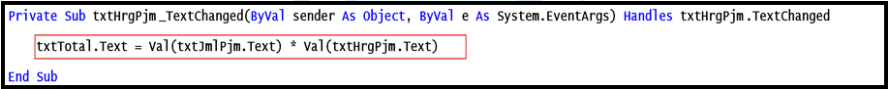
* Kemudian Ketikkan Program Berikut Pada Keypress (Kotak Merah)



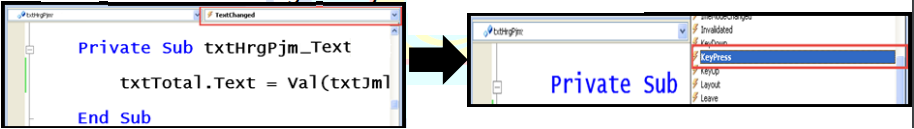
* Klik dua Kali TextBox Harga Pinjam Sekarang pada Desain FrmPinjamBuku



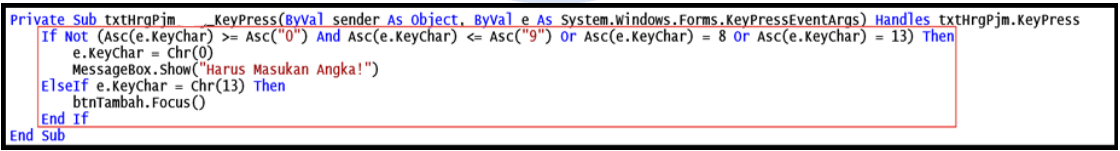
* Kemudian Ketikkan program berikut (Kotak Merah)



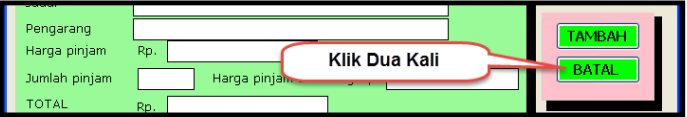
* Ubah Event TextChange Menjadi Keypress



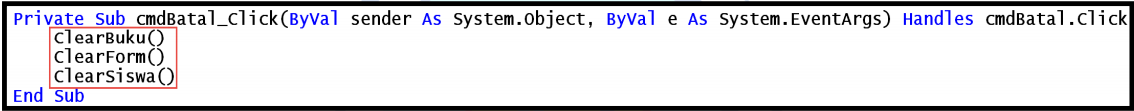
* Kemudian Ketikkan Program Berikut Pada Keypress (Kotak Merah)



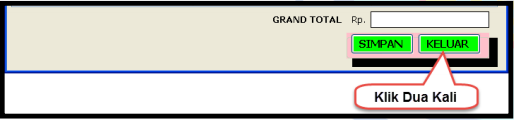
* Klik dua Kali Tombol BATAL pada Desain FrmPinjamBuku



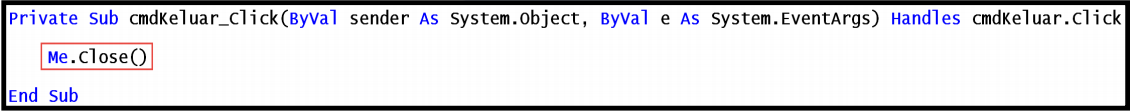
* Kemudian Ketikkan program berikut (Kotak Merah)



* Klik dua Kali Tombol KELUAR pada Desain FrmPinjamBuku



* Kemudian ketikkan program berikut (Kotak Merah)

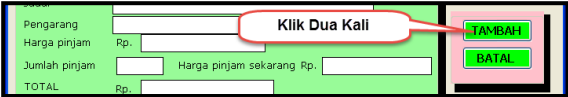


## 7.2. Menyimpan Data Peminjaman Buku

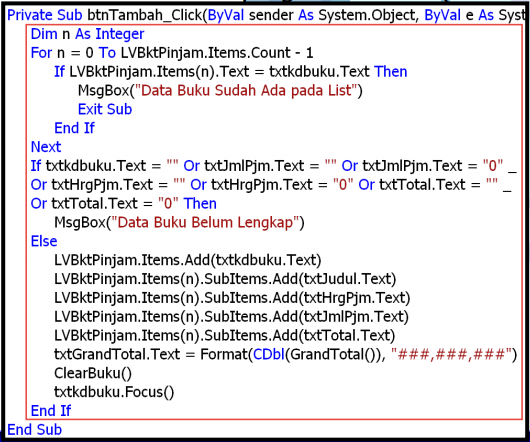
* Pada bagian awal, kita sudah membuat class untuk file transaksi.
* Pada bagian ini, akan dibahas menambah data ke Listview dan menyimpan transaksi ke dalam 2 table.
* Bila melihat ke database, maka ada 2 tabel yang perlu disimpan pada saat transaksi :

1. BuktiPinjam --> file transaksi
2. Ada --> file detil transaksi

* Klik dua Kali Tombol TAMBAH pada Desain FrmPinjamBuku



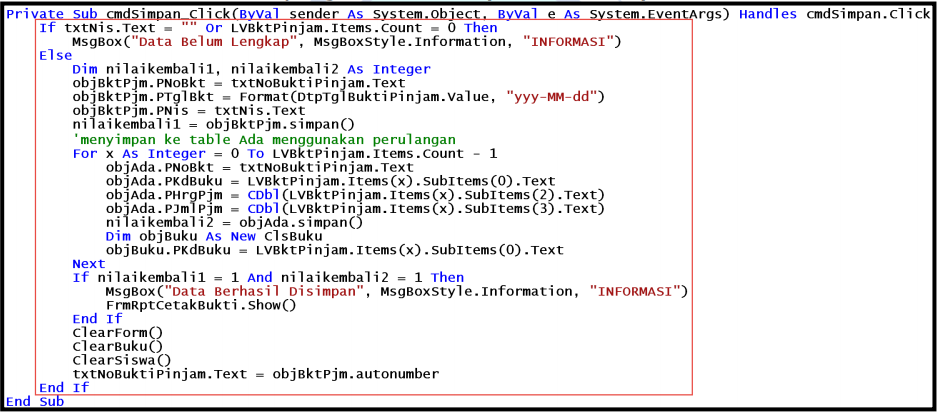
* Kemudian Ketikkan program berikut (Kotak Merah)



* Klik dua Kali Tombol BATAL pada Desain FrmPinjamBuku



* Kemudian Ketikkan program berikut (Kotak Merah)



## 9.1. Desain Form Master

Desainlah form stok seperti berikut ini:



Nama untuk setiap object pada gambar di atas:

A: txtKode

B: txtNama

C: txtJenis

D: txtJual

E: txtBeli

F: txtSisa

G: txt Lokasi

H: PictureBox1

I: btnBrowse

J: btnSimpan

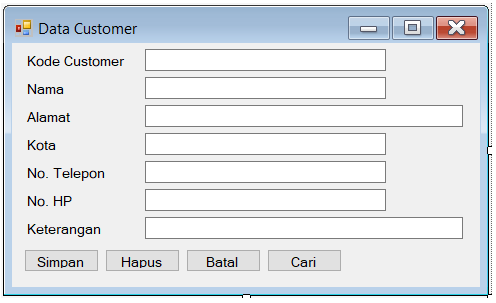
K: btnHapus

L: btnBatal

M: btnCari

Untuk textbox yang beriisikan informasi mengenai bilangan silahkan ubah property TextAlign menjadi Right.

Desainlah form customer seperti berikut ini:



Nama untuk setiap textbox secara berurutan dari atas ke bawah:

txtKode

txtNama

txtAlamat

txtKota

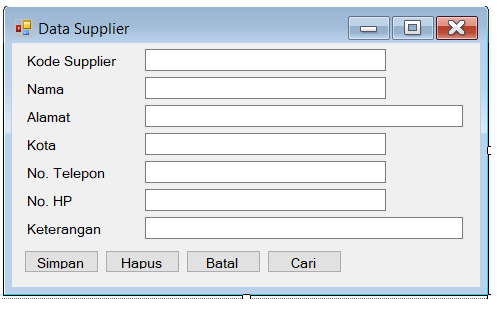
txtTelepon

txtHP

txtKeterangan

Untuk nama button sama seperti pada form stok sebelumnya.

Desainlah form supplier seperti berikut ini:



Nama untuk setiap textbox secara berurutan dari atas ke bawah:

txtKode

txtNama

txtAlamat

txtKota

txtTelepon

txtHP

txtKeterangan

Untuk nama button sama seperti pada form stok sebelumnya.

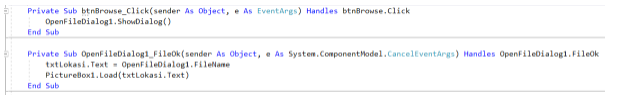
## 9.2. Source Code Form Stok

Berikut ini adalah code yang digunakan pada form stok (untuk tombol Cari akan dibahas pada pertemuan berikutnya):

Code berikut untuk inisialisasi awal seperti bersihkan setting koneksi dengan database, bersihkan form, dan code ketika form diload:



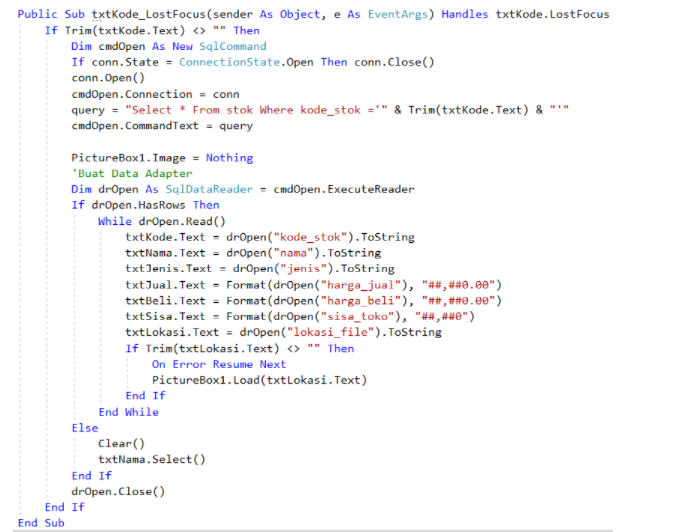
Code berikut berguna untuk mengisi gambar ke picture box:



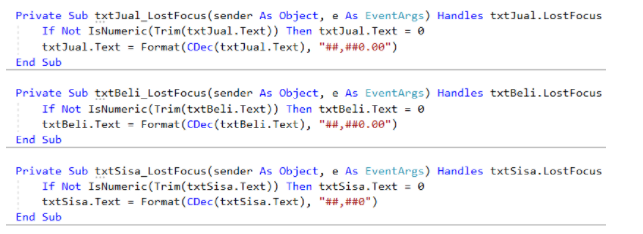
Code berikut berguna untuk menyimpan data:



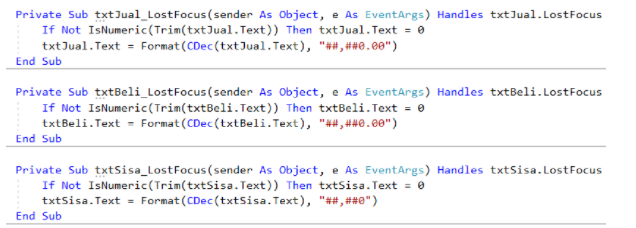
Code berikut berguna untuk menampilkan data ketika textbox Kode Stok terjadi lost focus:



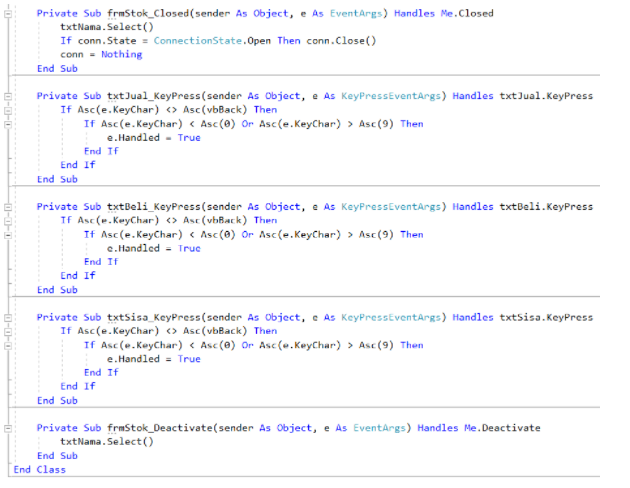
Code berikut berguna untuk tombol Batal dan Delete:



Code berikut untuk mensetting tampilan di informasi Harga Jual, Harga Beli, dan Sisa Stok:



Code berikut untuk menutup koneksi ketika form diclose, selain itu juga terdapat code untuk memvalidasi input pada harga jual, harga beli, dan sisa stok.



## 7.1. SELECT statement

SELECT statement memungkinkan  untuk memilih data dari satu atau lebih tabel. Berikut sintaksnya

SELECT select\_list

FROM table\_name;

Pada Sintaks tersebut :

* Pertama, tentukan satu atau lebih kolom darimana  ingin memilih data setelah SELECT keyword. Jika select\_list memiliki beberapa kolom,  memisahkannya dengan comma (,).
* Kedua, tentukan nama tabel darimana  ingin memilih data setelah FROM keyword.

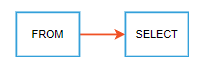
Titik koma (;) adalah opsional. Ini menunjukkan akhir dari sebuah pernyataan. Jika  memiliki dua atau lebih pernyataan,  perlu menggunakan titik koma (;) untuk memisahkannya sehingga MySQL akan mengeksekusi setiap pernyataan secara individual.

SELECT dan FROM adalah keywords. Dengan konvensi,  menulis kata kunci SQL dalam huruf besar. Namun, itu tidak wajib. Karena SQL tidak peka terhadap huruf besar/kecil, dapat menulis pernyataan SQL dalam huruf kecil, huruf besar, dll. Misalnya:

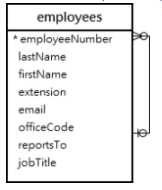
select select\_list

from table\_name;

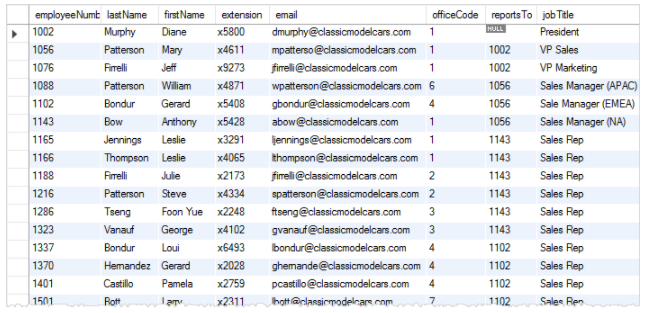
Saat mengeksekusi  SELECT statement, dievaluasi FROM clause sebelum SELECT clause:



## 7.2. Contoh

employees table pada [sample database](https://www.mysqltutorial.org/mysql-sample-database.aspx) untuk contoh berikut ini:  


employees tabel memiliki delapan kolom: employeeNumber, lastName, firstName, ekstensi, email, officeCode, reportTo, dan jobTitle. Tabel juga memiliki banyak baris seperti yang ditunjukkan pada gambar berikut:



A) Menggunakan pernyataan MySQL SELECT untuk mengambil data dari contoh kolom tunggal

Contoh berikut ini menggunakan SELECT statement untuk memilih nama belakang semua karyawan:

SELECT lastName

FROM employees;

Inilah sebagian outputnya:

+-----------+

| lastName |

+-----------+

| Murphy |

| Patterson |

| Firrelli |

| Patterson |

| Bondur |

| Bow |

| Jennings |

...

Hasil dari SELECT statement disebut kumpulan hasil karena merupakan kumpulan baris yang dihasilkan dari query.

Beberapa contoh lainnya terdapat pada link berikut [ini](https://www.mysqltutorial.org/tryit/query/mysql-select/#2), untuk dicoba

## 7.3. SELECT statement SELECT untuk meminta data dari beberapa kolom contoh

Contoh berikut ini menggunakan SELECT statement untuk mendapatkan nama depan, nama belakang, dan jabatan karyawan:

SELECT

lastName,

firstName,

jobTitle

FROM

employees;

Meskipun employees table memiliki banyak kolom,  SELECT statement mengembalikan data dari tiga kolom lastName, firstName, dan jobTitle ditentukan dalam SELECT clause:

+-----------+-----------+----------------------+

| lastname | firstname | jobtitle |

+-----------+-----------+----------------------+

| Murphy | Diane | President |

| Patterson | Mary | VP Sales |

| Firrelli | Jeff | VP Marketing |

| Patterson | William | Sales Manager (APAC) |

| Bondur | Gerard | Sale Manager (EMEA) |

## 7.4. SELECT statement untuk mengambil data dari semua kolom contoh

Jika  ingin memilih data dari semua kolom ini pada employees table, dapat menentukan semua nama kolom di SELECT clause seperti ini:

SELECT employeeNumber,

lastName,

firstName,

extension,

email,

officeCode,

reportsTo,

jobTitle

FROM employees;

Atau dapat menggunakan tanda (\*) yang merupakan singkatan untuk semua kolom. Sebagai contoh :

SELECT \*

FROM employees;

Query mengembalikan data dari semua kolom dari employees table.

SELECT \*  sering disebut “select star” or “select all” karena memilih data dari semua kolom tabel. Dalam praktiknya,  harus menggunakan SELECT \* atau query ad-hoc saja

Cobalah latihan pada link berikut [ini](https://www.mysqltutorial.org/tryit/query/mysql-select/#3)

## 7.5. SELECT DISTINCT Statement

SELECT DISTINCT statement digunakan untuk mengembalikan hanya nilai yang berbeda (berbeda).

Di dalam tabel, kolom sering kali berisi banyak nilai duplikat; dan terkadang  hanya ingin membuat daftar nilai yang berbeda (berbeda).

**SELECT DISTINCT Syntax**

SELECT DISTINCT column1, column2, ...  
FROM table\_name;

**SELECT Example Without DISTINCT**

Pernyataan SQL berikut memilih semua (termasuk duplikat) nilai dari kolom "Negara" pada "Customers" table:

**Contoh:**

SELECT Country FROM Customers;

Cobalah latihan pada link berikut [ini](https://www.w3schools.com/mysql/trymysql.asp?filename=trysql_select_no_distinct)

**SELECT DISTINCT Examples**

Pernyataan SQL berikut hanya memilih nilai DISTINCT dari kolom "Negara" pada "Customers" table:

**Contoh**

SELECT DISTINCT Country FROM Customers;

Cobalah latihan pada link berikut [ini](https://www.w3schools.com/mysql/trymysql.asp?filename=trysql_select_no_distinct)

Pernyataan SQL berikut menghitung dan mengembalikan jumlah negara yang berbeda (berbeda) pada "Customers" table:

**Contoh:**

SELECT COUNT(DISTINCT Country) FROM Customers;

Cobalah latihan pada link berikut [ini](https://www.w3schools.com/mysql/trymysql.asp?filename=trysql_select_distinct2)

Latihan pada link berikut [ini](https://www.w3schools.com/mysql/exercise.asp?filename=exercise_select2)

## Rangkuman Topik 7

Perintah SELECT digunakan untuk menampilkan kolom pada tabel. Bila menggunakan tanda \*, maka akan menampilkan seluruh isi tabel, namun bila ingin menampilkan kolom yang diinginkan, maka tanda \* diganti dengan nama kolom yang ingin ditampilkan. Hal tersebut dapat digunakan bila ingin menampilkan data pada tabel. Hal tersebut juga dapat digunakan pada 1 atau lebih dari 1 tabel, namun tetap berada pada 1 database. Sebagai bahan tambahan anda kerjakan latihan pada link berikut ini : [1](https://www.w3schools.com/mysql/exercise.asp?filename=exercise_select1), [2](https://www.w3schools.com/mysql/exercise.asp?filename=exercise_select2), [3](https://www.w3schools.com/mysql/exercise.asp?filename=exercise_select3)

## 9.1. Pengantar

WHERE clause memungkinkan  menentukan kondisi pencarian untuk baris yang dikembalikan oleh qeury. Berikut ini menunjukkan sintaks dari WHERE clause:

SELECT

select\_list

FROM

table\_name

WHERE

search\_condition;

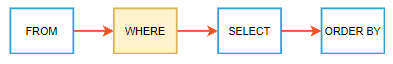
search\_condition adalah kombinasi dari satu atau lebih ekspresi menggunakan logika operator [AND](https://www.mysqltutorial.org/mysql-and/), [OR](https://www.mysqltutorial.org/mysql-or/) and NOT.

Di MySQL, predikat adalah ekspresi Boolean yang mengevaluasi ke TRUE, FALSE, or UNKNOWN.

SELECT statement akan menyertakan setiap baris yang memenuhi search\_condition dalam himpunan hasil.

Disamping SELECT statement, dapat menggunakan  WHERE clause pada [UPDATE](https://www.mysqltutorial.org/mysql-update-data.aspx) or [DELETE](https://www.mysqltutorial.org/mysql-delete-statement.aspx) statement untuk menentukan baris mana yang akan update or delete.

Saat menjalankan SELECT statement dengan  WHERE clause, MySQL mengevaluasi WHERE clause setelah FROM clause dan sebelum SELECT dan ORDER BY clauses:



## 9.2. WHERE clause with equality(=) operator dan AND operator

Kita akan menggunakab employees table dari [sample database](https://www.mysqltutorial.org/mysql-sample-database.aspx) untuk demonstrasi.

**1)  WHERE clause dengan contoh operator persamaan**

Query berikut  menggunakan  WHERE clause untuk menemukan semua karyawan yang jabatan pekerjaannya Sales Rep:

SELECT

lastname,

firstname,

jobtitle

FROM

employees

WHERE

jobtitle = 'Sales Rep';

+-----------+-----------+-----------+

| lastname | firstname | jobtitle |

+-----------+-----------+-----------+

| Jennings | Leslie | Sales Rep |

| Thompson | Leslie | Sales Rep |

| Firrelli | Julie | Sales Rep |

| Patterson | Steve | Sales Rep |

| Tseng | Foon Yue | Sales Rep |

| Vanauf | George | Sales Rep |

| Bondur | Loui | Sales Rep |

| Hernandez | Gerard | Sales Rep |

| Castillo | Pamela | Sales Rep |

| Bott | Larry | Sales Rep |

| Jones | Barry | Sales Rep |

| Fixter | Andy | Sales Rep |

| Marsh | Peter | Sales Rep |

| King | Tom | Sales Rep |

| Nishi | Mami | Sales Rep |

| Kato | Yoshimi | Sales Rep |

| Gerard | Martin | Sales Rep |

+-----------+-----------+-----------+

17 rows in set (0.00 sec)

SELECT statement memeriksa semua baris dari employees table dan hanya memilih baris yang nilainya dalam jobTitle kolom Sales Rep.

**2)  WHERE clause dengan AND operator**

* WHERE clause untuk menemukan karyawan yang jabatan pekerjaannya Sales Rep dan kode office adalah 1:

SELECT

lastname,

firstname,

jobtitle,

officeCode

FROM

employees

WHERE

jobtitle = 'Sales Rep' AND

officeCode = 1;

+----------+-----------+-----------+------------+

| lastname | firstname | jobtitle | officeCode |

+----------+-----------+-----------+------------+

| Jennings | Leslie | Sales Rep | 1 |

| Thompson | Leslie | Sales Rep | 1 |

+----------+-----------+-----------+------------+

2 rows in set (0.00 sec)

Ekpresi  WHERE clause menggunakan [AND](https://www.mysqltutorial.org/mysql-and/) operator untuk menggabungkan dua kondisi:

jobtitle = 'Sales Rep' AND officeCode = 1;

AND operator mengevaluasi untuk TRUE hanya jika kedua ekspresi bernilai TRUE. Oleh karena itu, query mengembalikan baris yang nilainya di jobTitle column adalah Sales Rep dan officeCode adalah 1.

* Pernyataan SQL berikut memilih semua bidang dari:  "Customers" Dimana negara adalah "Germany" AND kota adalah "Berlin":

SELECT \* FROM Customers  
WHERE Country = 'Germany' AND City = 'Berlin';  
   
[TRY](https://www.w3schools.com/mysql/trymysql.asp?filename=trysql_select_where_and)

## 9.3. WHERE clause dengan OR operator dan BETWEEN operator

### **WHERE clause with OR operator**

Query ini menemukan karyawan yang jabatannya adalah Sales Rep atau karyawan yang menemukan kantor dengan kode kantor 1:

SELECT

lastName,

firstName,

jobTitle,

officeCode

FROM

employees

WHERE

jobtitle = 'Sales Rep' OR

officeCode = 1

ORDER BY

officeCode ,

jobTitle;

+-----------+-----------+--------------------+------------+

| lastName | firstName | jobTitle | officeCode |

+-----------+-----------+--------------------+------------+

| Murphy | Diane | President | 1 |

| Bow | Anthony | Sales Manager (NA) | 1 |

| Jennings | Leslie | Sales Rep | 1 |

| Thompson | Leslie | Sales Rep | 1 |

| Firrelli | Jeff | VP Marketing | 1 |

| Patterson | Mary | VP Sales | 1 |

| Firrelli | Julie | Sales Rep | 2 |

| Patterson | Steve | Sales Rep | 2 |

| Tseng | Foon Yue | Sales Rep | 3 |

| Vanauf | George | Sales Rep | 3 |

| Bondur | Loui | Sales Rep | 4 |

| Hernandez | Gerard | Sales Rep | 4 |

| Castillo | Pamela | Sales Rep | 4 |

| Gerard | Martin | Sales Rep | 4 |

| Nishi | Mami | Sales Rep | 5 |

| Kato | Yoshimi | Sales Rep | 5 |

| Fixter | Andy | Sales Rep | 6 |

| Marsh | Peter | Sales Rep | 6 |

| King | Tom | Sales Rep | 6 |

| Bott | Larry | Sales Rep | 7 |

| Jones | Barry | Sales Rep | 7 |

+-----------+-----------+--------------------+------------+

21 rows in set (0.00 sec)

[OR](https://www.mysqltutorial.org/mysql-or/) operator mengevaluasi untuk TRUE hanya jika salah satu ekspresi bernilai TRUE:

jobtitle = 'Sales Rep' OR officeCode = 1

Oleh karena itu, kueri mengembalikan setiap karyawan yang memiliki jabatan Sales Rep atau kode kantor 1.

### **WHERE clause dengan BETWEEN operator**

[BETWEEN](https://www.mysqltutorial.org/mysql-between) operator kemblai TRUE jika suatu nilai berada dalam rentang nilai:

expression BETWEEN low AND high

Query berikut menemukan karyawan yang mencari di kantor yang kode kantornya dari 1 hingga 3:

SELECT

firstName,

lastName,

officeCode

FROM

employees

WHERE

officeCode BETWEEN 1 AND 3

ORDER BY officeCode;

+-----------+-----------+------------+

| firstName | lastName | officeCode |

+-----------+-----------+------------+

| Diane | Murphy | 1 |

| Mary | Patterson | 1 |

| Jeff | Firrelli | 1 |

| Anthony | Bow | 1 |

| Leslie | Jennings | 1 |

| Leslie | Thompson | 1 |

| Julie | Firrelli | 2 |

| Steve | Patterson | 2 |

| Foon Yue | Tseng | 3 |

| George | Vanauf | 3 |

+-----------+-----------+------------+

10 rows in set (0.00 sec)

## 9.4. WHERE clause with the LIKE operator dan IN operator

### **WHERE clause with the LIKE operator example**

[LIKE](https://www.mysqltutorial.org/mysql-like/) operator mengevaluasi untuk TRUE jika suatu nilai cocok dengan pola yang ditentukan.

Untuk membentuk pola, Anda menggunakan % dan \_ wildcards.  % wildcard cocok dengan string apapun dari nol atau lebih karakter saat \_ wildcard cocok dengan karakter tunggal apa pun.

Query berikut menemukan karyawan yang nama belakangnya diakhiri dengan string'son':

SELECT

firstName,

lastName

FROM

employees

WHERE

lastName LIKE '%son'

ORDER BY firstName;

+-----------+-----------+

| firstName | lastName |

+-----------+-----------+

| Leslie | Thompson |

| Mary | Patterson |

| Steve | Patterson |

| William | Patterson |

+-----------+-----------+

4 rows in set (0.00 sec)

### **WHERE clause with the IN operator example**

[IN](https://www.mysqltutorial.org/mysql-basics/mysql-in/) operator megembalikan TRUE jika suatu nilai cocok dengan nilai apapun dalam daftar.

value IN (value1, value2,...)

Contoh berikut menggunakan WHERE clause dengan IN operator untuk mencari karyawan yang berlokasi di kantor dengan kode kantor 1.

SELECT

firstName,

lastName,

officeCode

FROM

employees

WHERE

officeCode IN (1 , 2, 3)

ORDER BY

officeCode;

+-----------+-----------+------------+

| firstName | lastName | officeCode |

+-----------+-----------+------------+

| Diane | Murphy | 1 |

| Mary | Patterson | 1 |

| Jeff | Firrelli | 1 |

| Anthony | Bow | 1 |

| Leslie | Jennings | 1 |

| Leslie | Thompson | 1 |

| Julie | Firrelli | 2 |

| Steve | Patterson | 2 |

| Foon Yue | Tseng | 3 |

| George | Vanauf | 3 |

+-----------+-----------+------------+

10 rows in set (0.00 sec)

## 9.5. WHERE clause with the IS NULL operator dan comparison operators

### **WHERE clause with the IS NULL operator**

Untuk memeriksa apakah suatu nilai adalah [NULL](https://www.mysqltutorial.org/mysql-null/) atau tidak, Anda menggunakan [IS NULL](https://www.mysqltutorial.org/mysql-is-null/) operator, bukan operator yang sama (=).  IS NULL operator mengembalikan TRUE jika nilai  NULL.

value IS NULL

Di dunia basis data, NULL adalah penanda yang menunjukkan bahwa nilai hilang atau tidak diketahui. Dan NULL tidak setara dengan angka 0 atau string kosong.

Pernyataan berikut menggunakan WHERE clause dengan IS NULL operatoruntuk mendapatkan baris dengan nilai dalam reportsTo kolom Adalah  NULL:

SELECT

lastName,

firstName,

reportsTo

FROM

employees

WHERE

reportsTo IS NULL;

+----------+-----------+-----------+

| lastName | firstName | reportsTo |

+----------+-----------+-----------+

| Murphy | Diane | NULL |

+----------+-----------+-----------+

1 row in set (0.01 sec)

### **WHERE clause with comparison operators**

Tabel berikut menunjukkan operator perbandingan yang dapat Anda gunakan untuk membentuk ekspresi i WHERE clause.

| **Operator** | **Description** |
| --- | --- |
| = | Equal to. You can use it with almost any data type. |
| <> or != | Not equal to |
| < | Less than. You typically use it with numeric and date/time data types. |
| > | Greater than. |
| <= | Less than or equal to |
| >= | Greater than or equal to |

Query berikut menggunakan operator tidak sama dengan (<>) untuk menemukan semua karyawan yang bukan Sales Rep:

SELECT

lastname,

firstname,

jobtitle

FROM

employees

WHERE

jobtitle <> 'Sales Rep';

+-----------+-----------+----------------------+

| lastname | firstname | jobtitle |

+-----------+-----------+----------------------+

| Murphy | Diane | President |

| Patterson | Mary | VP Sales |

| Firrelli | Jeff | VP Marketing |

| Patterson | William | Sales Manager (APAC) |

| Bondur | Gerard | Sale Manager (EMEA) |

| Bow | Anthony | Sales Manager (NA) |

+-----------+-----------+----------------------+

6 rows in set (0.00 sec)

Query berikut menemukan karyawan yang kode kantornya lebih besar dari 5:

SELECT

lastname,

firstname,

officeCode

FROM

employees

WHERE

officecode > 5;

+-----------+-----------+------------+

| lastname | firstname | officeCode |

+-----------+-----------+------------+

| Patterson | William | 6 |

| Bott | Larry | 7 |

| Jones | Barry | 7 |

| Fixter | Andy | 6 |

| Marsh | Peter | 6 |

| King | Tom | 6 |

+-----------+-----------+------------+

6 rows in set (0.00 sec)

Query berikut mengembalikan karyawan dengan kode kantor kurang dari atau sama dengan 4 (<=4):

SELECT

lastname,

firstname,

officeCode

FROM

employees

WHERE

officecode <= 4;

+-----------+-----------+------------+

| lastname | firstname | officeCode |

+-----------+-----------+------------+

| Murphy | Diane | 1 |

| Patterson | Mary | 1 |

| Firrelli | Jeff | 1 |

| Bondur | Gerard | 4 |

| Bow | Anthony | 1 |

| Jennings | Leslie | 1 |

| Thompson | Leslie | 1 |

| Firrelli | Julie | 2 |

| Patterson | Steve | 2 |

| Tseng | Foon Yue | 3 |

| Vanauf | George | 3 |

| Bondur | Loui | 4 |

| Hernandez | Gerard | 4 |

| Castillo | Pamela | 4 |

| Gerard | Martin | 4 |

+-----------+-----------+------------+

15 rows in set (0.00 sec)

## 9.6. Combining AND, OR and NOT

Anda juga dapat menggabungkan AND, OR and NOT operators.

Pernyataan SQL berikut memilih semua bidang dari: "Customers" di mana negara adalah  "Germany" AND Kota adalah "Berlin" OR "Stuttgart" (gunakan tanda kurung untuk membentuk ekspresi kompleks):

**Contoh**

SELECT \* FROM Customers  
WHERE Country = 'Germany' AND (City = 'Berlin' OR City = 'Stuttgart');

[Try](https://www.w3schools.com/mysql/trymysql.asp?filename=trysql_select_where_and_or)

Pernyataan SQL berikut memilih semua bidang dari "Pelanggan" dimana negara NOT "Germany" dan NOT "USA":

**Contoh**

SELECT \* FROM Customers  
WHERE NOT Country = 'Germany' AND NOT Country = 'USA';

[Try](https://www.w3schools.com/mysql/trymysql.asp?filename=trysql_select_where_not_and)

**Test Yourself With Exercises  
Exercise:**

Pilih semua record di mana City column memiliki nilai 'Berlin' dan PostalCode column memiliki nilai 12209.

[Try](https://www.w3schools.com/mysql/exercise.asp?filename=exercise_where4)

## Rangkuman Topik 9

* Menggunakan WHERE clause untuk memfilter baris berdasarkan kondisi.
* MySQL mengevaluasi WHERE clause setelah FROM clause dan Sebelum SELECT dan ORDER BY clauses.
* WHERE clause dapat dikombinasikan AND, OR, and NOT operators.
* AND and OR operators digunakan untuk memfilter record berdasarkan lebih dari satu kondisi:
  + - AND operator menampilkan catatan jika semua kondisi dipisahkan oleh AND adalah TRUE.
    - OR operator menampilkan catatan jika ada kondisi yang dipisahkan oleh OR adalah TRUE.
    - NOT operator menampilkan catatan jika kondisinya adalah  NOT TRUE.

## 10.1. Introduction

INSERT statement memungkinkan  untuk memasukkan satu atau lebih baris ke dalam tabel. Berikut ini mengilustrasikan sintaks dari INSERT statement:

INSERT INTO table(c1,c2,...)

VALUES (v1,v2,...);

Dalam sintaks ini,

* Pertama, tentukan nama tabel dan daftar kolom yang dipisahkan koma di dalam tanda kurung setelah INSERT INTO clause.
* Kemudian, letakkan daftar nilai yang dipisahkan koma dari kolom yang sesuai di dalam tanda kurung mengikuti VALUES keyword.

Jumlah kolom dan nilai harus sama. Selain itu, posisi kolom harus sesuai dengan posisi nilainya.

Untuk [insert multiple rows](https://www.mysqltutorial.org/mysql-insert-multiple-rows/) ke dalam tabel menggunakan single INSERT statement, syntax nya sebagai berikut:

INSERT INTO table(c1,c2,...)

VALUES

(v11,v12,...),

(v21,v22,...),

...

(vnn,vn2,...);

Dalam sintaks ini, baris dipisahkan dengan koma di VALUES clause.

Untuk menyimpan data kedalam database [MySQL](https://www.eplusgo.com/mysql/) digunakan perintah INSERT.

//Digunakan apabila Anda menginput data dengan kolom yang tidak spesifik  
INSERT INTO nama\_tabel VALUES (isian1, isian2, isian3,…);

atau

//Digunakan apabila Anda menginput data pada kolom tertentu  
INSERT INTO nama\_tabel (kolom1, kolom2, kolom3,…)  
VALUES (isian1, isian2, isian3,…);

Contoh :

|  |
| --- |
| INSERT INTO user (nama\_depan, nama\_belakang, pesan) VALUES ('Muhammad','Fariz','Ayo belajar'); |

## 10.2. Contoh

Buatlah [create a new table](https://www.mysqltutorial.org/mysql-create-table/) dengan nama tasks untuk latihan INSERT statement.

CREATE TABLE IF NOT EXISTS tasks (

task\_id INT AUTO\_INCREMENT,

title VARCHAR(255) NOT NULL,

start\_date DATE,

due\_date DATE,

priority TINYINT NOT NULL DEFAULT 3,

description TEXT,

PRIMARY KEY (task\_id)

);

## 10.2.1. Simple INSERT example

Pernyataan berikut menyisipkan baris baru ke dalam tasks table:

INSERT INTO tasks(title,priority)

VALUES('Learn MySQL INSERT Statement',1);

MySQL mengembalikan pesan berikut:

1 row(s) affected

Ini berarti bahwa satu baris telah dimasukkan ke dalam  tasks table successfully.

Query ini mengembalikan data dari tasks table:

SELECT \* FROM tasks;

output:

Dalam contoh ini, hanya ditentukan nilai  untuk title dan priority columns. Untuk kolom lainnya, MySQL menggunakan nilai default.

task\_id column adalah [AUTO\_INCREMENT](https://www.mysqltutorial.org/mysql-sequence/) column. Ini berarti bahwa MySQL menghasilkan integer sekuensial setiap kali baris dimasukkan ke dalam tabel.

start\_date, due\_date, dan description columns menggunakan NULL sebagai nilai default, oleh karena itu, MySQL menggunakan [NULL](https://www.mysqltutorial.org/mysql-null/) untuk menyisipkan ke dalam kolom ini jika Anda tidak menentukan nilainya di  INSERT statement.

## 10.2.2. Memasukkan baris menggunakan contoh nilai default

Jika  ingin memasukkan nilai default ke dalam kolom, terdapat dua cara:

* Abaikan nama kolom dan nilai di INSERT statement.
* Tentukan nama kolom di INSERT INTO clause dan gunakan DEFAULT

kata kunci di VALUES clause.

Contoh berikut menunjukkan cara kedua:

INSERT INTO tasks(title,priority)

VALUES('Understanding DEFAULT keyword in INSERT statement',DEFAULT);

Dalam contoh ini, kami menentukan priority column dan  DEFAULT keyword.

Karena nilai default untuk prioritas kolom adalah 3 seperti yang dinyatakan dalam definisi tabel:

priority TINYINT NOT NULL DEFAULT 3

MySQL menggunakan angka 3 untuk dimasukkan ke dalam priority column.

Pernyataan berikut mengembalikan isi dari tasks table setelah insert:

SELECT \* FROM tasks;

## 10.2.3. Memasukkan tanggal ke dalam contoh tabel

Untuk menyisipkan nilai tanggal literal ke dalam kolom, Anda menggunakan format berikut:

'YYYY-MM-DD'

Format:

* YYYY mewakili tahun empat digit e.g., 2018.
* MM mewakili bulan dua digit e.g., 01, 02, and 12.
* DD mewakili hari dua digit e.g., 01, 02, 30.

Pernyataan berikut menyisipkan baris baru ke tasks table dengan nilai tanggal mulai dan jatuh tempo:

INSERT INTO tasks(title, start\_date, due\_date)

VALUES('Insert date into table','2018-01-09','2018-09-15');

Gambar berikut menunjukkan isi dari tasks table setelah insert:

Dimungkinkan untuk menggunakan ekspresi dalam VALUES clause.

Misalnya, pernyataan berikut menambahkan [tugas](http://spada.unprimdn.ac.id/mod/assign/view.php?id=438204) baru menggunakan tanggal saat ini untuk kolom tanggal mulai dan tanggal jatuh tempo:

INSERT INTO tasks(title,start\_date,due\_date)

VALUES('Use current date for the task',CURRENT\_DATE(),CURRENT\_DATE())

Contoh menggunakan CURRENT\_DATE() function sebagai nilai untuk  
 start\_date and due\_date columns. Perhatikan bahwa  
 [CURRENT\_DATE()](https://www.mysqltutorial.org/mysql-curdate/) function adalah [date function](https://www.mysqltutorial.org/mysql-date-functions/) yang mengembalikan tanggal sistem saat ini.

Berikut adalah isi dari tasks table setelah insert:

## 10.2.4. Contoh menyisipkan beberapa baris

Pernyataan berikut menyisipkan tiga baris ke dalam tasks table:

INSERT INTO tasks(title, priority)

VALUES

('My first task', 1),

('It is the second task',2),

('This is the third task of the week',3);

Dalam contoh ini, setiap data baris ditetapkan sebagai daftar nilai dalam VALUES clause.

MySQL mengembalikan pesan berikut:

3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0

Ini berarti bahwa tiga baris telah berhasil dimasukkan tanpa duplikat atau peringatan.

SELECT \* FROM tasks;

 tasks table menampilkan data berikut:

## Rangkuman Topik 10

INSERT INTO statement ada 2 cara:

1. Tentukan nama kolom yang akan disisipkan.

INSERT INTO table\_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);

2. Jika Anda menambahkan nilai untuk semua kolom tabel, Anda tidak perlu menentukan nama kolom dalam kueri SQL. Namun, pastikan urutan nilai dalam urutan yang sama dengan kolom dalam tabel. Di sini, sintaks INSERT INTO adalah sebagai berikut:

INSERT INTO table\_name  
VALUES (value1, value2, value3, ...);

## 11.1. INSERT multiple rows statement

Untuk insert beberapa baris/record kedalam sebuah tabel, gunakan  INSERT statement berikut ini:

INSERT INTO table\_name (column\_list)

VALUES

(value\_list\_1),

(value\_list\_2),

...

(value\_list\_n);

syntax:

* + Pertama, tentukan nama tabel yang ingin Anda sisipkan setelah INSERT INTO keywords.
  + Kedua, tentukan daftar kolom yang dipisahkan koma di dalam tanda kurung setelah nama tabel
  + Ketiga, tentukan daftar data baris yang dipisahkan koma di VALUES clause. Setiap elemen daftar mewakili satu baris. Jumlah nilai pada setiap elemen harus sama dengan jumlah kolom pada column\_list.

## 11.2. INSERT multiple rows limit

Secara teori, anda dapat insert beberapa baris/record menggunakan single INSERT statement. Namun, ketika server MySQL menerima INSERT statement yang ukurannya lebih besar dari max\_allowed\_packet, akan menimbulkan kesalahan yang terlalu besar dan mengakhiri koneksi.

Pernyataan ini menunjukkan nilai saat ini dari max\_allowed\_packet variable:

SHOW VARIABLES LIKE 'max\_allowed\_packet';

Jalankan perintah tersebut, lalu perhatikan tampilan pada server dengan berikut ini, apakah sama?

Value column adalah tope data bytes.

Untuk menetapkan nilai baru untuk max\_allowed\_packet variable, gunakan statement:

SET GLOBAL max\_allowed\_packet=size;

size adalah integer yang mewakili jumlah ukuran paket maksimum yang diizinkan dalam byte.

Perhatikan bahwa max\_allowed\_packet tidak memiliki pengaruh pada [INSERT INTO .. SELECT](https://www.mysqltutorial.org/mysql-insert-into-select/) statement.  INSERT INTO .. SELECT statement dapat insert sebanyak baris yang diinginkan.

## 11.3. INSERT multiple rows example

Contoh  INSERT  statement menggunakan banyak baris.

Pertama, [create a new table](https://www.mysqltutorial.org/mysql-create-database/) disebut projects untuk didemonstrasikan:

CREATE TABLE projects(

project\_id INT AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

start\_date DATE,

end\_date DATE,

PRIMARY KEY(project\_id)

);

Kedua, gunakan INSERT multiple rows statement untuk insert two baris ke projects table:

INSERT INTO

projects(name, start\_date, end\_date)

VALUES

('AI for Marketing','2019-08-01','2019-12-31'),

('ML for Sales','2019-05-15','2019-11-20');

Menampilkan pesan sebagai berikut:

2 row(s) affected

Berhasil menampilkan 2 baris yang di inster ke projects table.

Perhatikan ketika  insert multiple rows dan gunakan  [LAST\_INSERT\_ID()](https://www.mysqltutorial.org/mysql-last_insert_id.aspx) function untuk mendapatkan  terakhir dimasukkan ke [AUTO\_INCREMENT](https://www.mysqltutorial.org/mysql-sequence/) column, akan mendapatkan id dari baris yang dimasukkan pertama saja, bukan id dari baris yang dimasukkan terakhir.

Ketiga, gunakan [SELECT](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) statement untuk memverifikasi insert:

SELECT \* FROM projects;

output:

## 11.4. INSERT INTO SELECT

Selain menggunakan nilai baris dalam VALUES clause, dapat menggunakan [SELECT](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) statement sebagai  data source untuk INSERT statement.

Berikut ilustrasi INSERT INTO SELECT statement:

INSERT INTO table\_name(column\_list)

SELECT

select\_list

FROM

another\_table

WHERE

condition;

Sintaks VALUES clause, menggunakan SELECT statement.  SELECT statement dapat mengambil data dari satu atau lebih tabel.

INSERT INTO SELECT statement sangat berguna ketika Anda ingin menyalin data dari tabel lain ke tabel atau meringkas data dari beberapa tabel ke dalam tabel.

## 11.5. NSERT INTO SELECT example

Pertama, [create a new table](https://www.mysqltutorial.org/mysql-create-table/) disebut dengan suppliers :

CREATE TABLE suppliers (

supplierNumber INT AUTO\_INCREMENT,

supplierName VARCHAR(50) NOT NULL,

phone VARCHAR(50),

addressLine1 VARCHAR(50),

addressLine2 VARCHAR(50),

city VARCHAR(50),

state VARCHAR(50),

postalCode VARCHAR(50),

country VARCHAR(50),

customerNumber INT,

PRIMARY KEY (supplierNumber)

);

Eksekusi perintah untuk membuat  suppliers table.

Misalkan semua pelanggan dari California, USA menjadi pemasok perusahaan. Berikut query menemukan semua pelanggan yang berlokasi di California, USA:

SELECT

customerNumber,

customerName,

phone,

addressLine1,

addressLine2,

city,

state,

postalCode,

country

FROM

customers

WHERE

country = 'USA' AND

state = 'CA';



Kedua, gunakan INSERT INTO ... SELECT statement untuk insert customers yang berlokasi di California USA dari  customers table ke   suppliers table:

INSERT INTO suppliers (

supplierName,

phone,

addressLine1,

addressLine2,

city,

state,

postalCode,

country,

customerNumber

)

SELECT

customerName,

phone,

addressLine1,

addressLine2,

city,

state ,

postalCode,

country,

customerNumber

FROM

customers

WHERE

country = 'USA' AND

state = 'CA';

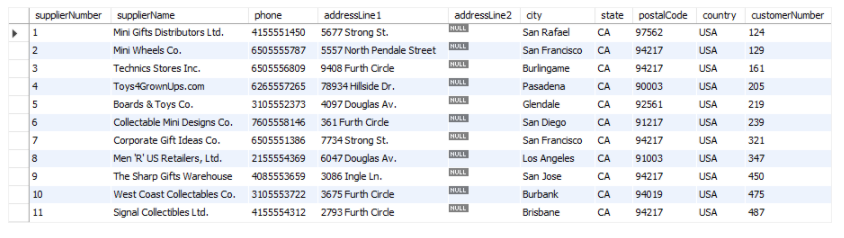
Menampilkan pesan yang menunjukkan bahwa 11 rows berhasil tampil.

11 row(s) affected Records: 11 Duplicates: 0 Warnings: 0

Ketiga, verifikasi penyisipan dengan menanyakan data dari suppliers table:

SELECT \* FROM suppliers;

output:



## 11.6. Using SELECT statement in the VALUES list

Pertama, [create a new table](https://www.mysqltutorial.org/mysql-create-table/) dengan nama stats:

CREATE TABLE stats (

totalProduct INT,

totalCustomer INT,

totalOrder INT

);

Kedua, gunakan [INSERT](https://www.mysqltutorial.org/mysql-insert-statement.aspx) statement untuk insert values yang berasal dari SELECT statements:

INSERT INTO stats(totalProduct, totalCustomer, totalOrder)

VALUES(

(SELECT COUNT(\*) FROM products),

(SELECT COUNT(\*) FROM customers),

(SELECT COUNT(\*) FROM orders)

);

 example:

* Pertama, gunakan SELECT statements denganCOUNT() functions unttuk mendapatkan:  total products, employees, and orders.
* Kedua, gunakan nilai yang dikembalikan dari [SELECT](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) statement menggantikan nilai-nilai dalam VALUES clause pada [INSERT](https://www.mysqltutorial.org/mysql-insert-statement.aspx) statement.

Ketiga, query data dari  table stats:

SELECT \* FROM stats;

## Rangkuman Topik 11

Query adalah menampilkan data yang dinginkan dari suatu data/tabel. Bila ingin mengetahui informasi tertentu dari satu atau lebih tabel, gunakanlah query. Misal: ingin mengetahu mahasiswa prodi Teknik Informatikan yang berdomisi kota medan, maka kolom yang ditampilkan hanyalah: NIM, Nama, prodi dan alamat. Sehingga membantu mempermudah untuk melihat seluruh informasi yang diinginkan. Bayangkan saja, bila anda menampilkan isi seluruh tabel, maka akan kesulitan menemukan domisili, apalagi bila record pada data tersebut berjumlah ribuan(dataset/big data)