

# **MODUL MATA KULIAH PRAKTIKUM BASIS DATA**

## **PROGRAM STUDI TEKNIK INFORMATIKA**

Tim Penulis:

Ashri Shabrina Afrah, M.T

Tri Mukti Lestari, M.Kom

Nur Fitriyah Ayu Tunjung Sari, M.Cs

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**FAKULTAS SAINS DAN TEKNOLOGI**  
**UIN MAULANA MALIK IBRAHIM MALANG**  
**2022**

## MODUL 1

### PENGENALAN MYSQL

#### 1.1 Bahasan dan Tujuan

##### 1.1.1 Bahasan

Membahas tentang pengertian basis data secara umum, pemahaman umum tentang MySQL, serta dasar-dasar pengoperasian basis data.

##### 1.1.2 Tujuan

1. Mahasiswa memahami pengertian basis data secara umum dan peranannya.
2. Mahasiswa memahami kegunaan software MySQL sebagai Database Management System (DBMS)
3. Mahasiswa memahami langkah-langkah instalasi software MySQL
4. Mahasiswa memahami cara mengkonfigurasi basis data MySQL

#### 1.2 Dasar Teori

##### 1.2.1 Basis data

Basis data terdiri dari 2 kata, yaitu basis dan data. Basis berarti markas/gudang atau tempat berkumpul. Sedangkan data merupakan representasi fakta dunia nyata yang mewakili suatu objek yang diwujudkan dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya (Fathansyah, 2018). Basis data dapat diartikan sebagai kumpulan data yang saling berhubungan yang disimpan secara bersama tanpa pengulangan (redundansi) yang tidak perlu supaya dapat dimanfaatkan kembali dengan cepat dan mudah.

Basis data tidak diolah secara manual, melainkan ditangan dengan menggunakan sebuah perangkat lunak yang disebut Database Management System (DBMS). Fungsi dari DBMS adalah untuk menyimpan, mengorganisasi, dan menggunakan kembali data di dalam basis data. Pada DBMS juga terdapat mekanisme untuk pengamanan data dan pemakaian bersama (*sharing*) dari data.

Pada DBMS, data yang telah disimpan dapat diakses dengan perintah-perintah tertentu. Perintah-perintah yang digunakan untuk mengelola basis data mempunyai standar yang disebut dengan SQL (*Structured QueryLanguage*). Standar ini dibuat oleh suatu badan yang berwenang (ANSI) sehingga sering disebut juga dengan istilah ANSI SQL. Saat ini standar SQL yang diacu kebanyakan software

adalah SQL92 dan SQL99. Pada umumnya, data yang tersimpan merupakan data relasional (data yang saling terhubung).

### 1.2.2 Tabel

Pada basis data relasional, data disimpan di dalam sejumlah tabel 2 dimensi. Setiap tabel terdiri atas lajur vertikal yang disebut *column/field* dan lajur mendatar yang disebut dengan *row/record*. Sebuah *field* menggambarkan atribut/karakteristik dari data, sedangkan setiap record adalah nilai dari atribut tersebut (atau dapat dikatakan sebagai data itu sendiri).

kode_MK	nama_MK	sks
20231	Basis Data	3
20232	Algoritma dan Pemrograman	3
20233	Pancasila	2

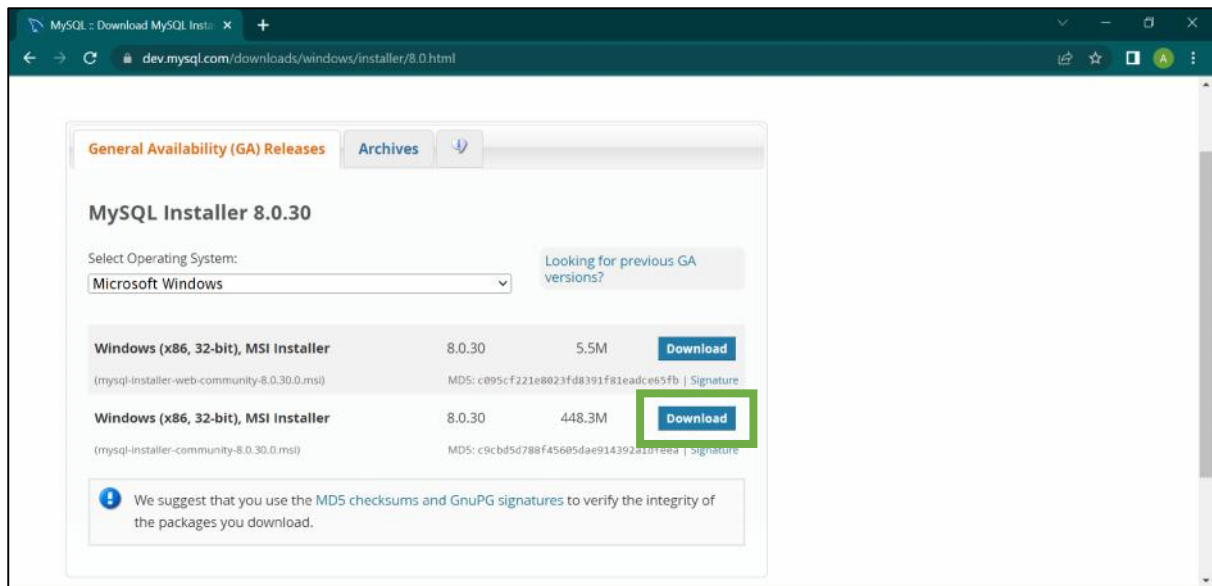
### 1.2.3 MySQL

MySQL adalah turunan dari SQL (Structured Query Language). SQL adalah sebuah konsep pengoperasian basis data, terutama untuk proses seleksi, pemasukan, perubahan, dan penghapusan data. Beberapa keunggulan dari MySQL adalah (Sutiaji, 2012):

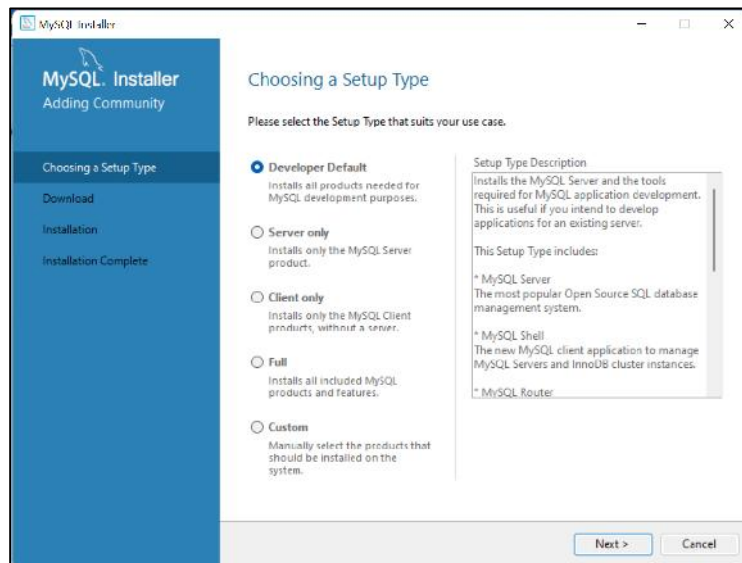
1. Dapat berjalan stabil pada berbagai sistem operasi.
2. Dapat digunakan oleh beberapa user secara bersamaan.
3. Memiliki kecepatan tinggi untuk menjalankan query.
4. Mampu menangani basis data dalam jumlah besar.
5. Memiliki struktur tabel yang lebih fleksibel dalam menangani alter table dibandingkan dengan PostgreSQL dan Oracle.

### 1.3 Instalasi MySQL

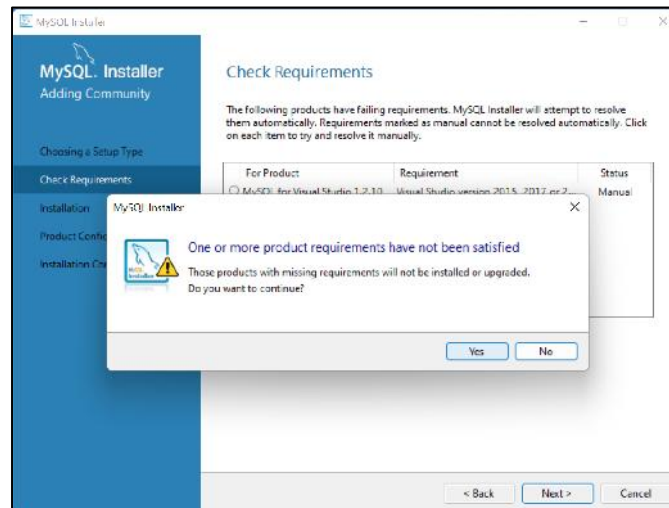
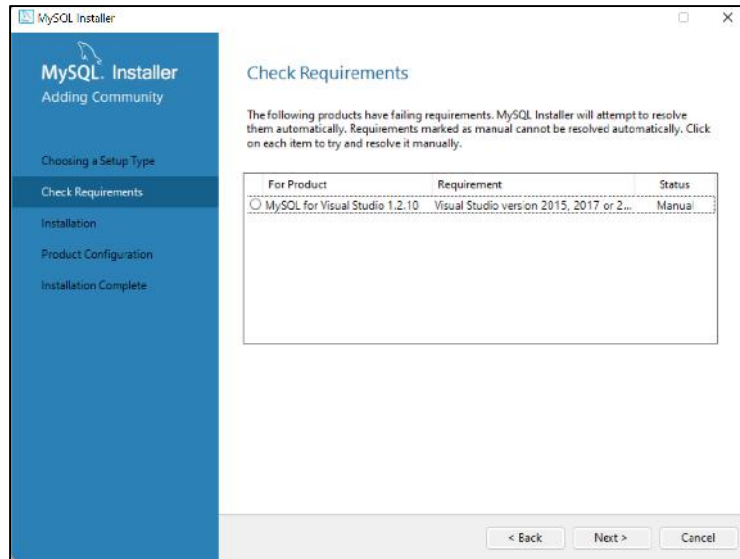
1. Siapkan file installer MySQL bertipe .msi yang dapat diunduh pada link <http://dev.mysql.com/downloads/mysql/>



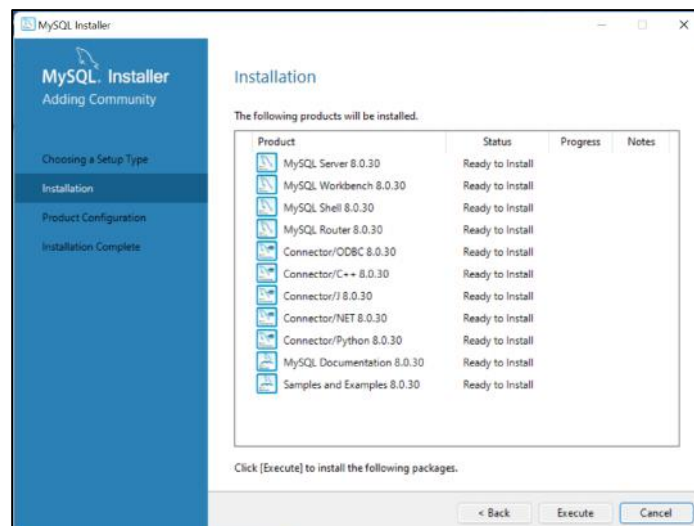
2. Double-click pada file mysql-installer-community-8.0.30.0.msi. Pada jendela MySQL Installer, pilih tipe setup **Developer Default**. Klik Next.

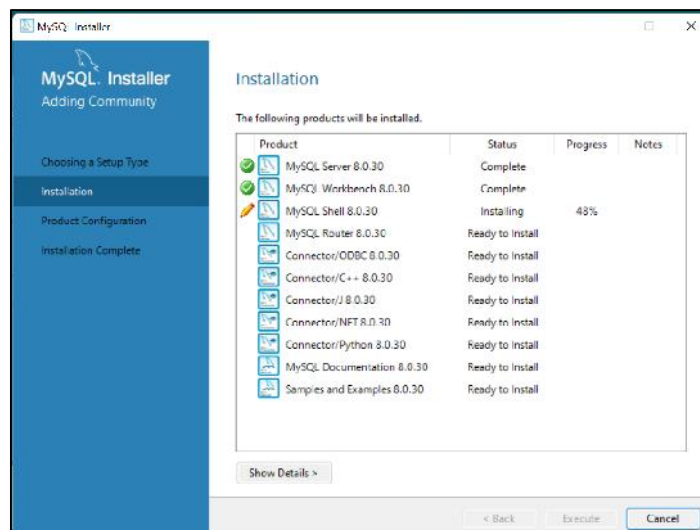


3. Pada langkah selanjutnya, klik Next. Akan muncul peringatan bahwa ada kebutuhan (requirements) dari proses instalasi yang belum terinstal. Klik Yes untuk tetap menginstal software MySQL.

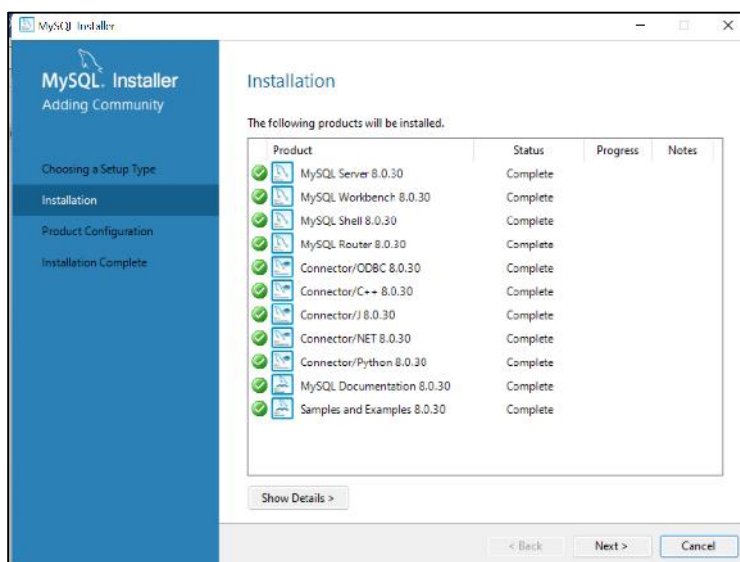


4. Selanjutnya kita akan masuk pada tahap instalasi. Click Execute.

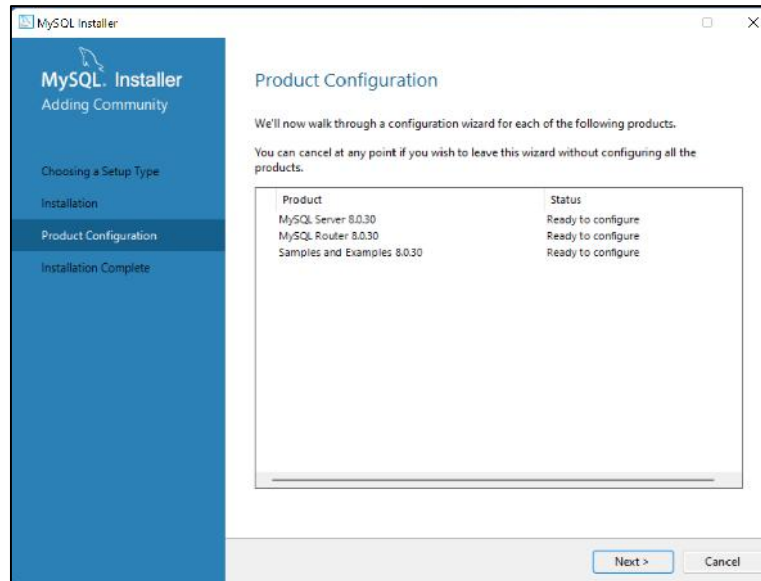




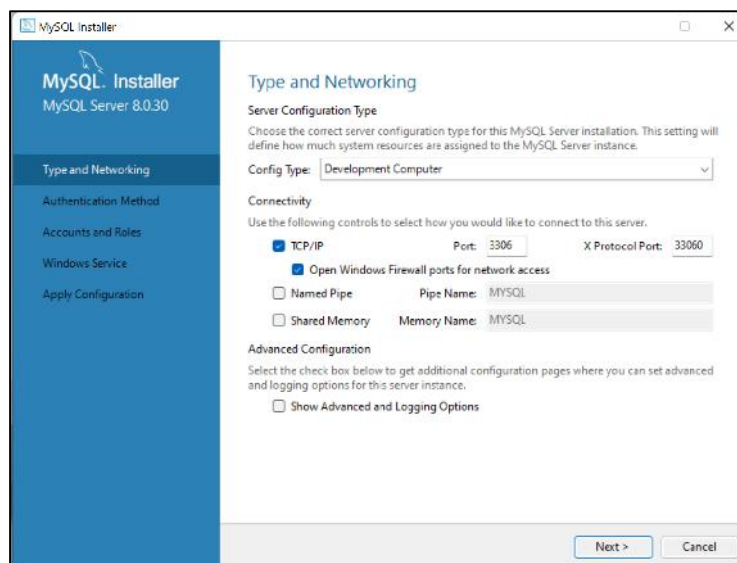
5. Setelah proses instalasi selesai. Klik Next



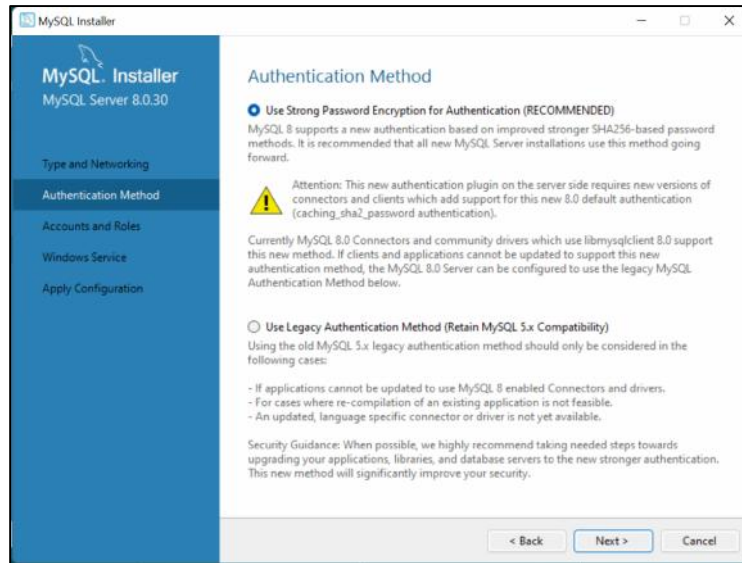
6. Selanjutnya, kita akan melakukan tahap konfigurasi. Klik Next.



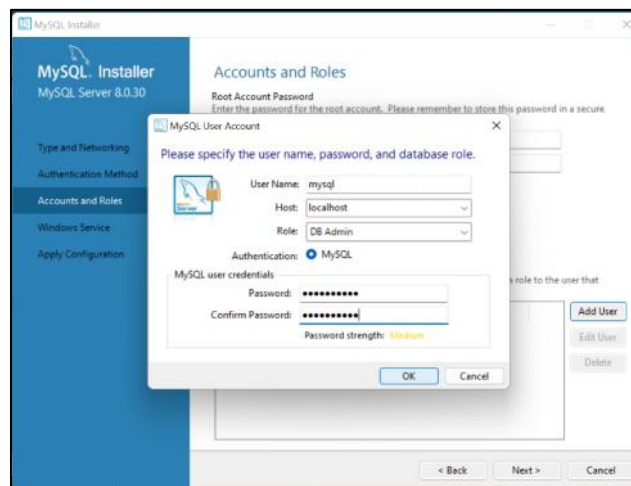
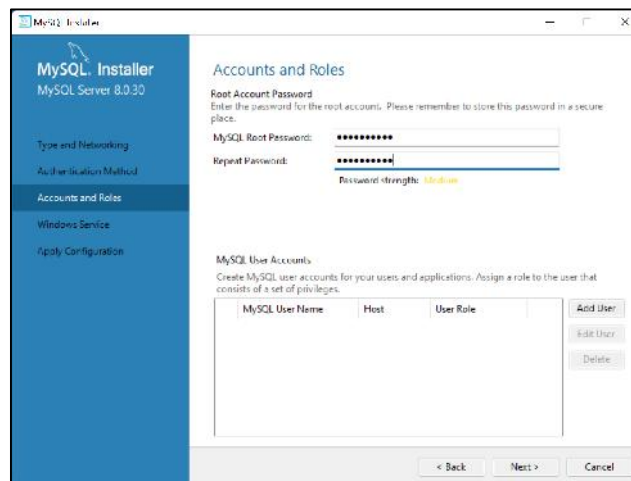
7. Pada tampilan selanjutnya, pilih Development Computer pada Config Type. Pilih TCP/IP dan port 3306 untuk metode koneksi server. Klik Next.



8. Selanjutnya kita akan melakukan pengaturan autentikasi. Pilih opsi **Use Strong Password Encryption for Authentication**. Klik Next.

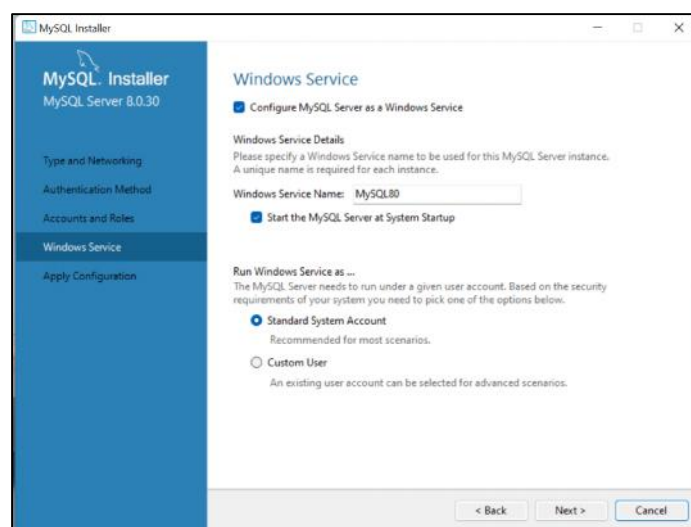


9. Atur password untuk akun root dari MySQL. Pastikan password yang digunakan unik dan mudah diingat. Kita juga dapat menambahkan user lain dengan meng-klik tombol Add User.

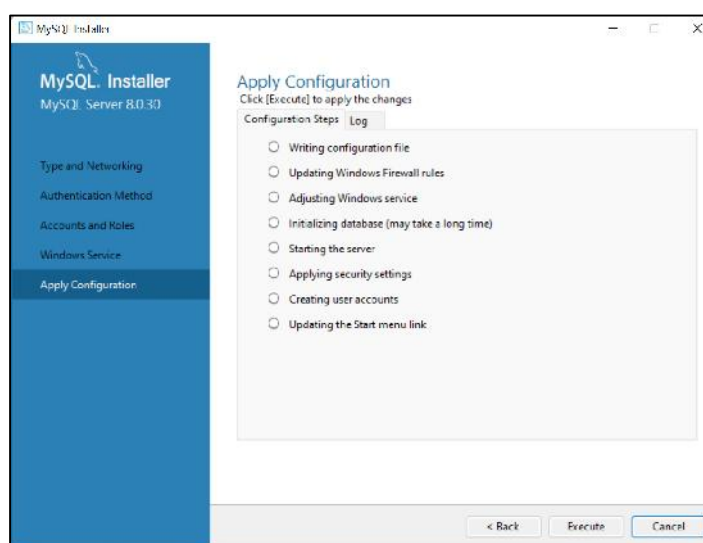




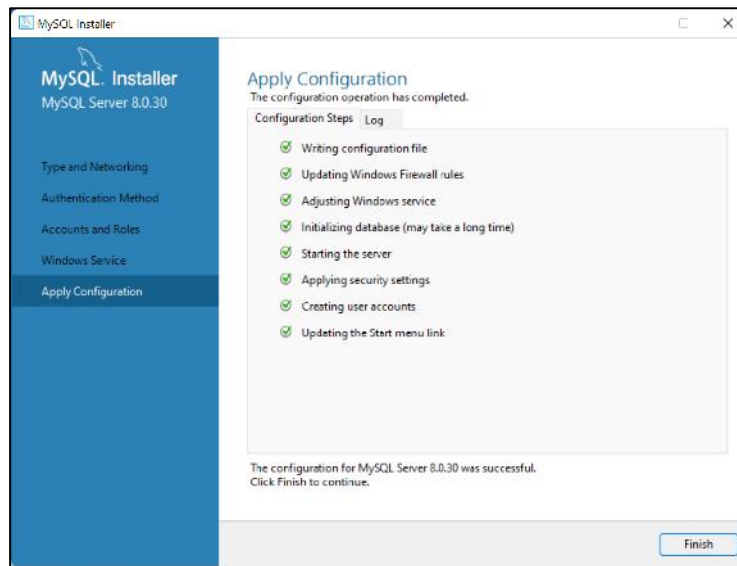
10. Kita akan masuk ke pengaturan Window Service seperti di bawah ini. Klik Next.



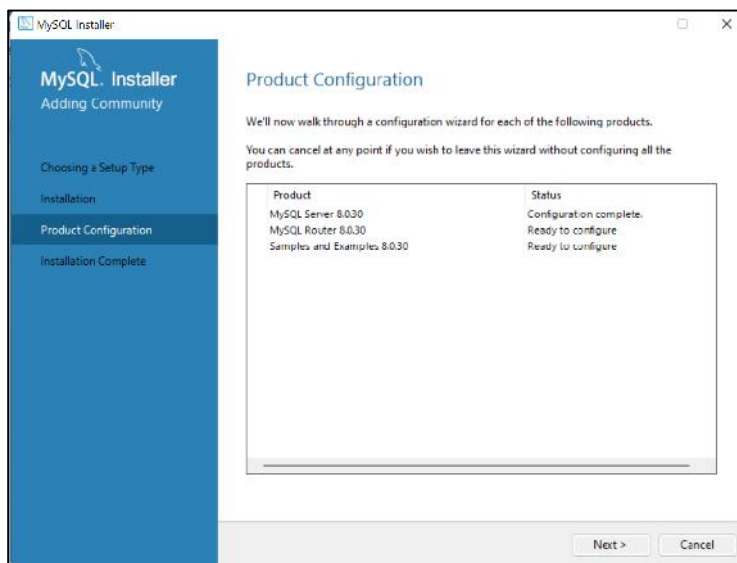
11. Selanjutnya kita akan mengaplikasikan konfigurasi yang telah dilakukan. Klik Execute.



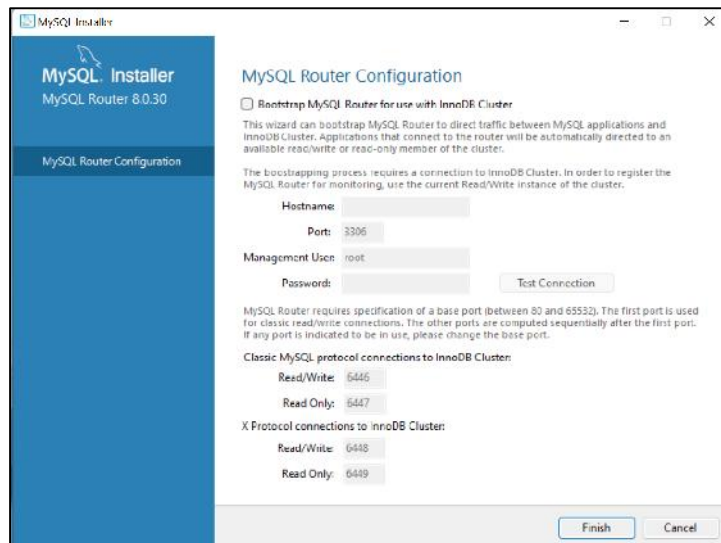
12. Setelah proses selesai. Klik Finish.



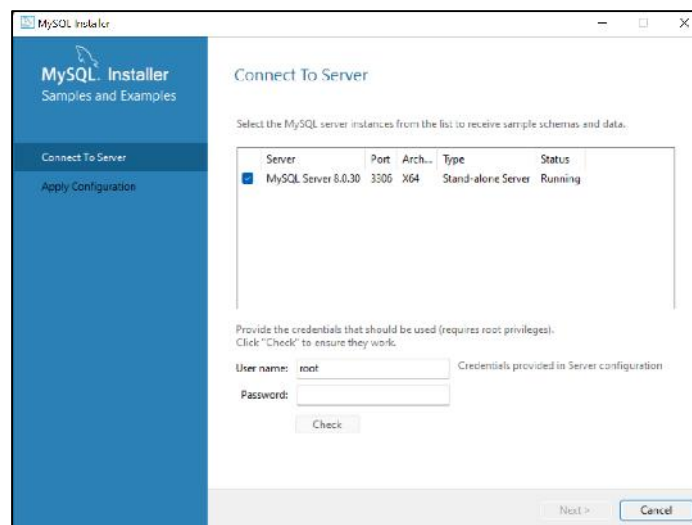
13. Langkah selanjutnya adalah melakukan konfigurasi pada MySQL Router.



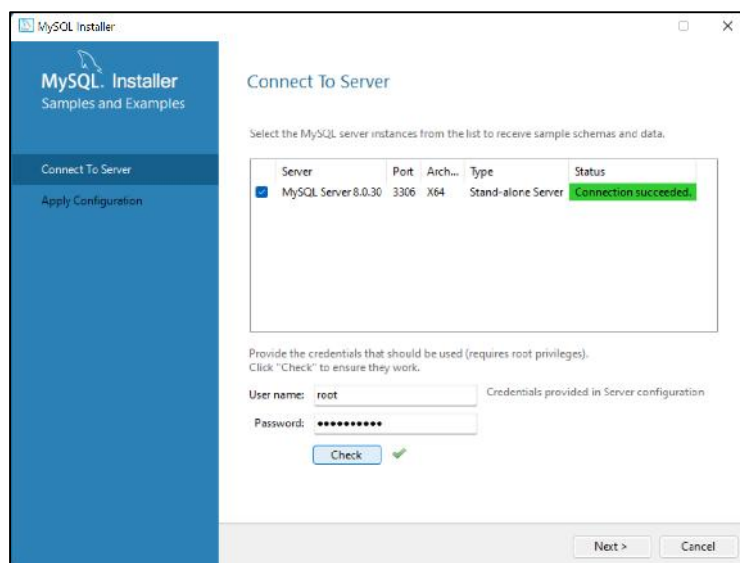
14. Jangan ubah pengaturan yang sudah dipilih secara default. Klik Finish.



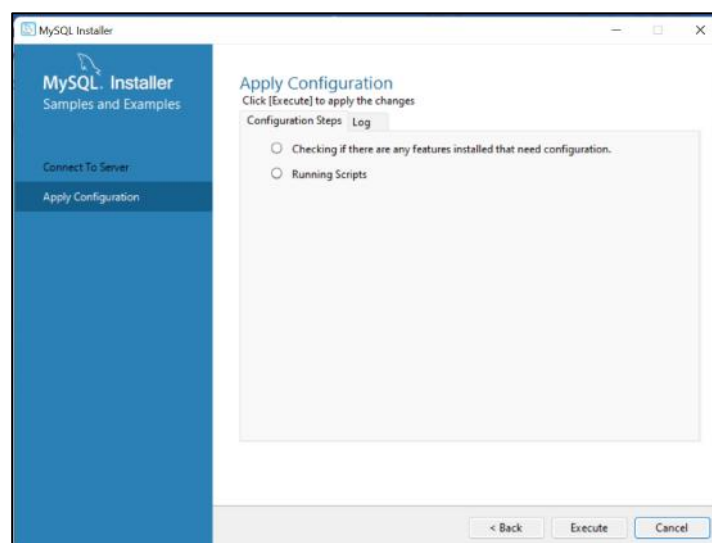
15. Selanjutnya, lakukan koneksi ke server. Masukkan password yang sudah dibuat untuk akun root pada langkah ke-9 yang sudah dilakukan sebelumnya. Klik Check.

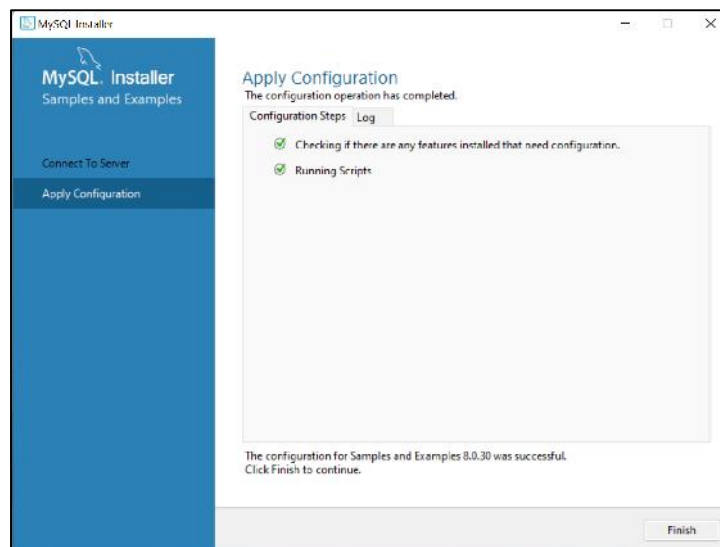


Apabila koneksi telah berhasil, klik Next.

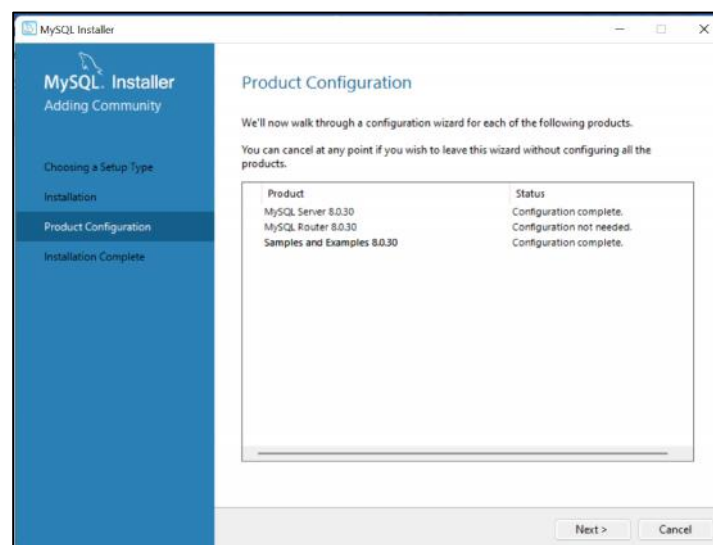


16. Pada tampilan selanjutnya, klik Execute untuk mengaplikasikan konfigurasi. Setelah konfigurasi selesai, klik Finish.

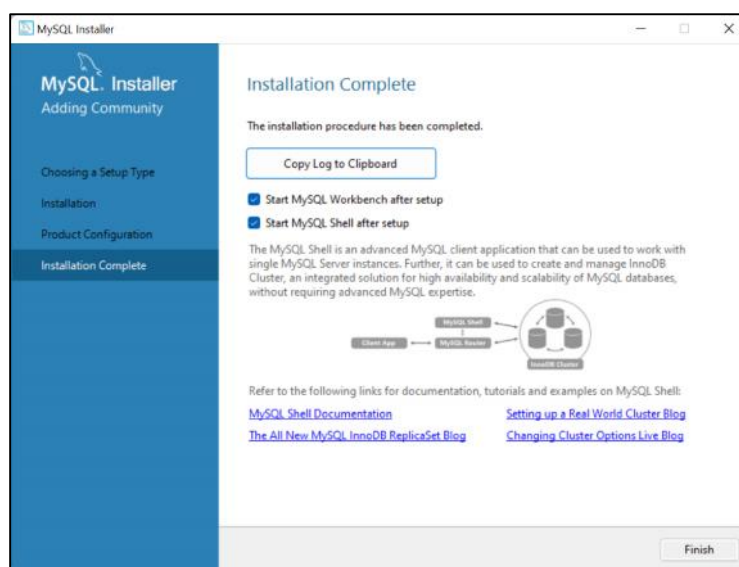




17. Proses konfigurasi telah selesai. Klik Next.



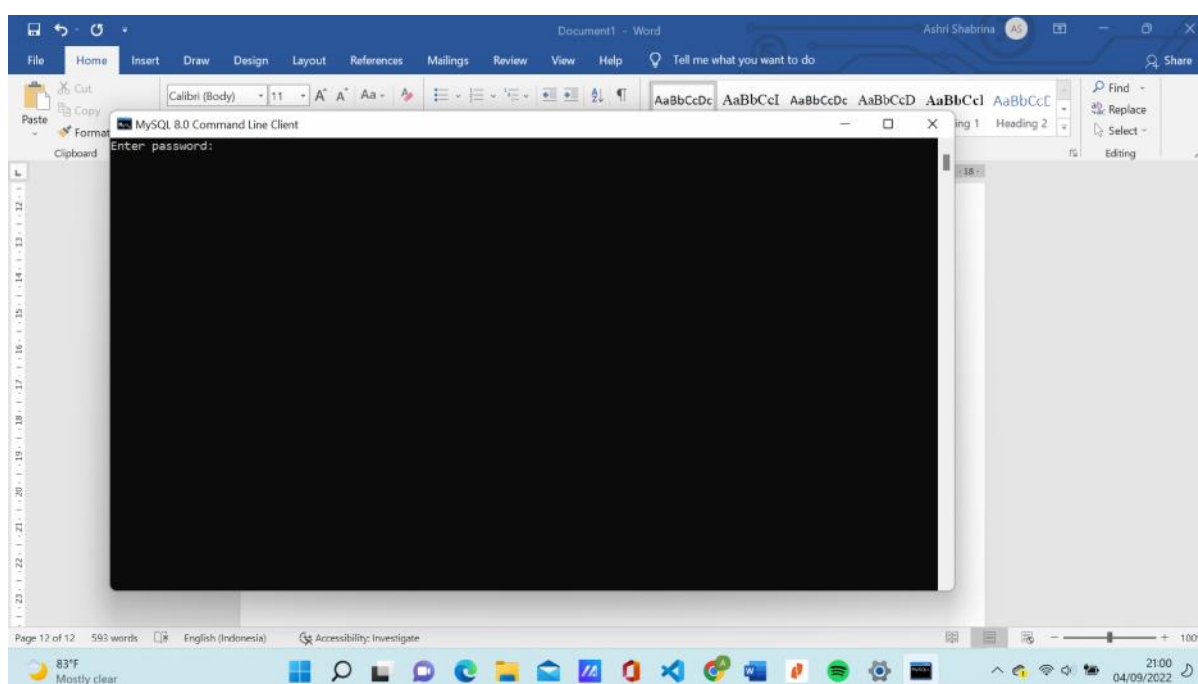
15. Proses instalasi telah selesai. Klik Finish.



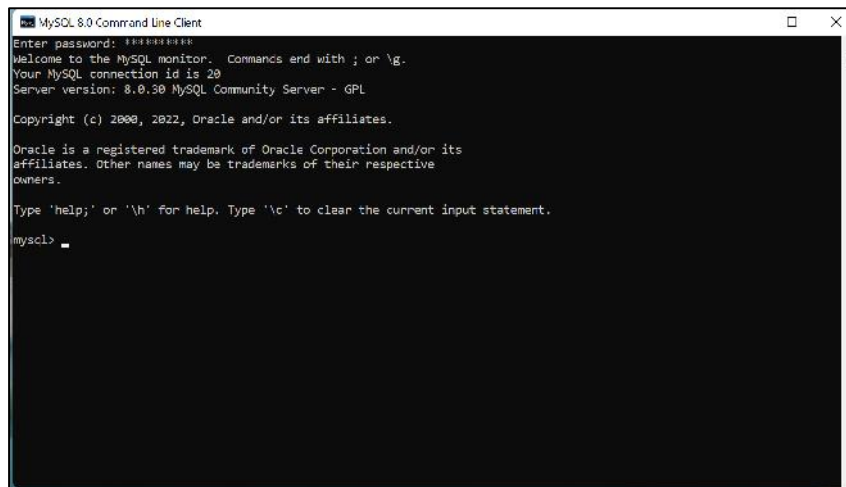
## 1.4 Membuat Basis Data Baru

### 1.4.1 Perintah-perintah pada MySQL 8.0 Command Line

#### 1. Buka MySQL 8.0 Command Line Client



#### 2. Masukkan password untuk akun root yang sudah dibuat pada proses instalasi.



```
MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 29
Server version: 8.0.30 MySQL Community Server - GPL

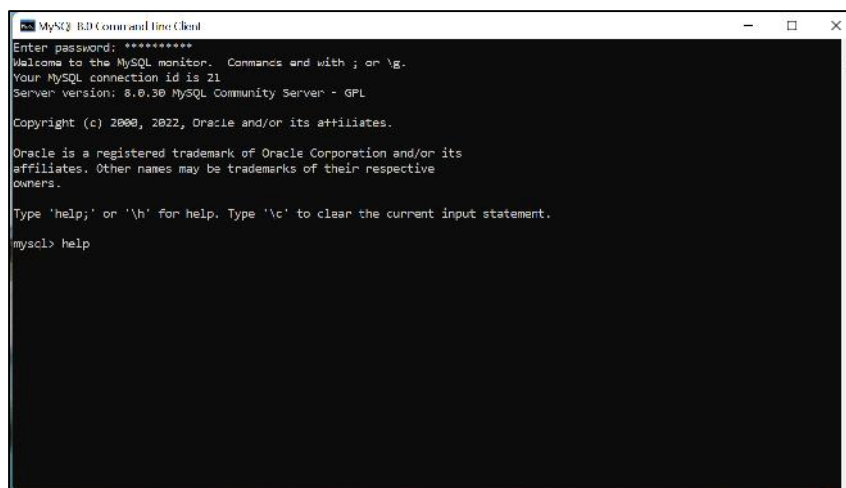
Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql>
```

3. Pada command line, kita dapat memasukkan berbagai perintah untuk menjalankan fungsi tertentu. Untuk melihat daftar perintah yang dapat digunakan, ketikkan **help**.



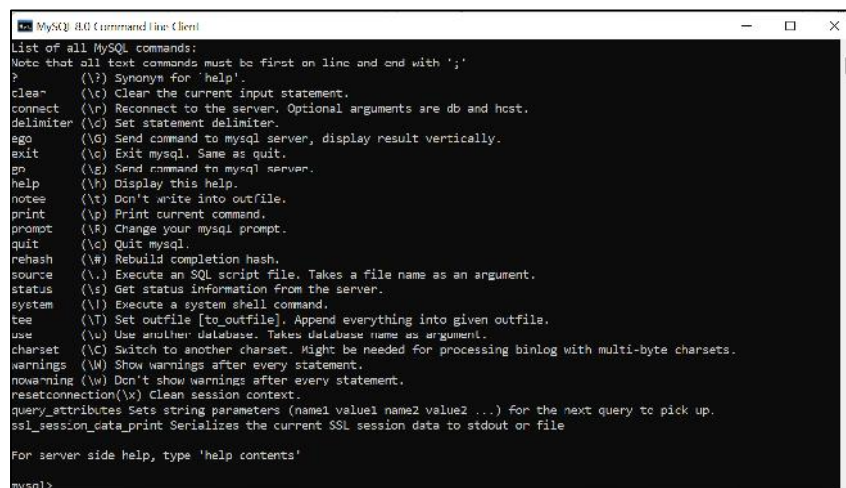
```
MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 31
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> help
```

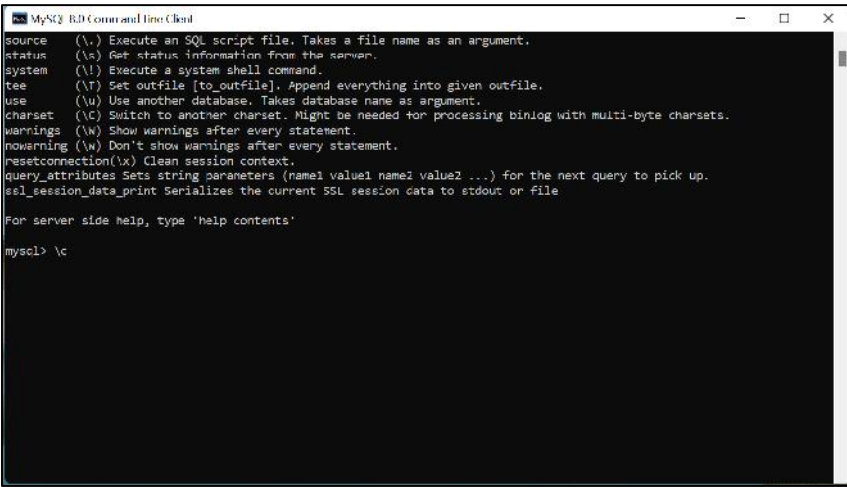


```
MySQL 8.0 Command Line Client
List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?          (\?) Synonym for 'help'.
clear     (\c) Clear the current input statement.
connect   (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set statement delimiter.
ego       (\g) Send command to mysql server, display result vertically.
exit      (\e) Exit mysql. Same as quit.
go        (\g) Send command to mysql server.
help      (\h) Display this help.
noexec    (\n) Don't write into outfile.
print     (\p) Print current command.
prompt    (\P) Change your mysql prompt.
quit      (\q) Quit mysql.
rehash   (\#) Rebuild completion hash.
source    (\.) Execute an SQL script file. Takes a file name as argument.
status    (\s) Get status information from the server.
system    (\!) Execute a system shell command.
tee       (\T) Set outfile [to_outfile]. Append everything into given outfile.
use       (\u) Use another database. Takes database name as argument.
charset   (\C) Switch to another charset. Might be needed for processing binlog with multi-byte charsets.
warnings  (\W) Show warnings after every statement.
warning   (\w) Don't show warnings after every statement.
resetconnection (\X) Clean session context.
query_attributes Sets string parameters (name1 value1 name2 value2 ...) for the next query to pick up.
ssl_session_data_print Serializes the current SSL session data to stdout or file

For server side help, type 'help contents'

mysql>
```

4. Untuk menutup command line, ketikkan **quit** atau **\q**.



```
MySQL RD Command Line Client
source (\.) Execute an SQL script file. Takes a file name as an argument.
status (\s) Get status information from the server.
system (\!) Execute a system shell command.
tee (\T) Set outfile [to outfile]. Append everything into given outfile.
use (\u) Use another database. Takes database name as argument.
charset (\C) Switch to another charset. Might be needed for processing binlog with multi-byte charsets.
warnings (\W) Show warnings after every statement.
nowarning (\NW) Don't show warnings after every statement.
resetconnection(\x) Clean session context.
query_attributes Sets string parameters (name1 value1 name2 value2 ...) for the next query to pick up.
ssl_session_data_print Serializes the current SSL session data to stdout or file

For server side help, type 'help contents'

mysql> \q
```

#### 1.4.2 Perintah CREATE DATABASE

Untuk membuat basis data baru, kita dapat menggunakan perintah Structured Query Language (SQL). Perintah SQL yang digunakan adalah sebagai berikut:

**CREATE DATABASE *nama\_basis\_data*;**

Pada perintah SQL di atas, ***nama\_basis\_data*** diisi dengan nama basis data yang akan kita buat. Nama basis data dapat terdiri dari huruf, angka, dan karakter khusus, namun tidak dapat dimulai dengan angka. Penulisan perintah SQL tidak *case sensitive*.

**Contoh:**

**CREATE DATABASE *toko\_1*;**



```
MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 25
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> CREATE DATABASE toko_1;
```

```
mysql> CREATE DATABASE toko_1;
Query OK, 1 row affected (0.00 sec)

mysql> _
```

Untuk mengaktifkan basis data yang sudah dibuat, kita dapat menggunakan perintah **USE**:

**USE toko\_1;**

```
mysql> USE toko_1;
Database changed
mysql>
```

### **1.5 TUGAS**

Buatlah laporan praktikum untuk mendokumentasikan langkah-langkah praktikum yang telah dilakukan. Sertakan gambar dari masing-masing langkah. Kumpulkan laporan dalam format PDF dengan penamaan file:

<nama>\_<NIM>\_<kelas dalam huruf>\_Praktikum 1.pdf

Contoh: Ananda\_20040567001\_A\_Praktikum1.pdf

## MODUL 2

### OBJEK BASIS DATA

#### 1.1 Bahasan dan Tujuan

##### 1.1.1 Bahasan

Membahas tentang objek-objek yang terdapat pada basis data, relasi antar tabel, dan penggunaan *relational key*.

##### 1.1.2 Tujuan

1. Mahasiswa mampu memahami objek-objek yang terdapat di dalam basis data.
2. Mahasiswa mampu menciptakan tabel di dalam basis data.
3. Mahasiswa mampu memahami penggunaan *relational key*.
4. Mahasiswa mampu membangun relasi antar tabel.

#### 1.2 Dasar Teori

##### 1.2.1 Objek Basis Data

a. Basis Data

Basis data terdiri dari 2 kata, yaitu basis dan data. Basis berarti markas/gudang atau tempat berkumpul. Sedangkan data merupakan representasi fakta dunia nyata yang mewakili suatu objek yang diwujudkan dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya (Fathansyah, 2018). Basis data dapat diartikan sebagai tempat berkumpulnya data yang saling berhubungan yang disimpan secara bersama tanpa pengulangan (redundansi) yang tidak perlu supaya dapat dimanfaatkan kembali dengan cepat dan mudah.

b. Tabel

Basis data relasional divisualisasikan dalam bentuk tabel yang terdiri dari sejumlah baris dan kolom untuk dapat menjelaskan hubungan data yang ada di dalamnya.

c. *Field*

*Field* adalah kolom yang ada di dalam tabel. *Field* disebut juga dengan atribut. Sebuah tabel terdiri dari beberapa atribut.

d. *Record*

*Record* adalah sebuah baris dalam suatu tabel. *Record* disebut juga dengan *row* atau *tupel*. Sebuah tabel dapat terdiri dari beberapa *record*.

e. *Primary Key*

*Primary key* adalah salah satu *relational key* yang digunakan sebagai pembeda antara *record* satu dengan yang lain. *Relational key* ditunjukkan dalam suatu atribut di dalam tabel. Selain *primary key* terdapat juga *relational key* lainnya seperti *Super Key*, *Candidate Key*, *Alternate Key*, dan *Foreign Key*.

f. *Relationship*

*Relationship* adalah hubungan antara beberapa tabel. Terdapat beberapa jenis relasi antara tabel di dalam basis data. Jenis-jenis relasi tersebut yaitu *One to One*, *One to Many*, dan *Many to Many*.

Tabel di bawah ini memberikan gambaran atau ilustrasi dari objek-objek yang ada di dalam basis data.

The diagram shows a table with three columns: **kode\_MK**, **nama\_MK**, and **sks**. The first column is underlined and labeled as the *primary key*. The rows are labeled as *record / baris / tupel*. The columns are collectively labeled as *field / kolom / atribut*.

<u>kode_MK</u>	nama_MK	sks
20231	Basis Data	3
20232	Algoritma dan Pemrograman	3
20233	Pancasila	2

## 1.3 Latihan Praktikum

### 1.3.1 Membuat Basis Data

Di dalam MySQL, basis data direpresentasikan sebagai sebuah folder yang didalamnya terdapat himpunan file-file tabel yang terdefinisi. Untuk membuat basis data baru, kita menggunakan perintah *Structured Query Language* (SQL). Perintah SQL yang digunakan adalah sebagai berikut:

```
CREATE DATABASE nama_basis_data;
```

Langkah-langkah untuk membuat basis data baru adalah sebagai berikut:

1. Buka *MySQL 8.0 Command Line Client* dan mulai membuat basis data baru dengan nama universitas. Nama basis data dapat terdiri dari huruf, angka, dan karakter khusus, namun tidak dapat dimulai dengan angka. Penulisan perintah SQL tidak *case sensitive*.

```
MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> CREATE DATABASE universitas;
Query OK, 1 row affected (0.04 sec)

mysql>
```

2. Untuk mengaktifkan basis data yang sudah dibuat, kita dapat menggunakan perintah **USE**:

```
mysql> USE universitas;
Database changed
mysql>
```

3. Apabila ingin melihat basis data yang sedang aktif, gunakan perintah **SELECT DATABASE()** seperti contoh di bawah ini:

```
mysql> SELECT DATABASE();
+-----+
| DATABASE() |
+-----+
| universitas |
+-----+
1 row in set (0.00 sec)
```

4. Sedangkan apabila ingin menampilkan daftar nama basis data yang ada di dalam DBMS, gunakan perintah **SHOW DATABASES**.

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| modul_1 |
| mysql |
| performance_schema |
| sakila |
| sys |
| universitas |
| world |
+-----+
8 rows in set (0.13 sec)
```

### 1.3.2 Membuat Tabel

Bentuk SQL pembuatan tabel baru adalah sebagai berikut:

```
CREATE TABLE nama_tabel(
  atribut 1 tipe_data(ukuran_atribut),
  atribut 2 tipe_data(ukuran_atribut),
  ....,
  atribut n tipe_data(ukuran_atribut),
  PRIMARY KEY(atribut_unik)
);
```

Ikuti langkah-langkah berikut untuk menciptakan tabel baru.

1. Menciptakan tabel mahasiswa dengan menggunakan SQL seperti di bawah ini.

```
mysql> CREATE TABLE mahasiswa(  
-> nim varchar(12),  
-> nama varchar(20),  
-> jk char(1),  
-> alamat varchar(20),  
-> PRIMARY KEY(nim)  
-> );  
Query OK, 0 rows affected (0.07 sec)
```

2. Apabila ingin melihat daftar nama tabel yang ada di dalam basis data, gunakan perintah **SHOW TABLES**.

```
mysql> SHOW TABLES;  
+-----+  
| Tables_in_universitas |  
+-----+  
| mahasiswa              |  
+-----+  
1 row in set (0.13 sec)
```

3. Sedangkan apabila ingin melihat struktur tabel yang sudah dibuat gunakan perintah **DESCRIBE nama tabel** atau **DESC nama tabel** seperti pada gambar di bawah ini.

```
mysql> DESCRIBE mahasiswa;  
+-----+-----+-----+-----+-----+-----+  
| Field | Type          | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| nim   | varchar(12)  | NO   | PRI | NULL    |       |  
| nama  | varchar(20)  | YES  |     | NULL    |       |  
| jk    | char(1)      | YES  |     | NULL    |       |  
| alamat | varchar(20)  | YES  |     | NULL    |       |  
+-----+-----+-----+-----+-----+-----+  
4 rows in set (0.10 sec)
```

### 1.3.3 Menambahkan *Record* di dalam Tabel

Menambahkan baris/*record*/tupel di dalam sebuah tabel dapat dilakukan dengan perintah **INSERT INTO**. Sebagai contoh yaitu menambahkan *record* baru di dalam tabel mahasiswa dengan SQL sebagai berikut:

```
mysql> INSERT INTO mahasiswa (nim,nama,jk,alamat) VALUES (200605110039,"Vera Artanti","P","JL. Sunan Muria 7");  
Query OK, 1 row affected (0.04 sec)
```

Apabila ingin menampilkan *record* yang berhasil ditambahkan dapat menggunakan perintah **SELECT**, seperti contoh di bawah ini.

```
mysql> SELECT * FROM mahasiswa;
+-----+-----+-----+-----+
| nim      | nama      | jk  | alamat      |
+-----+-----+-----+-----+
| 200605110039 | Vera Artanti | P    | JL. Sunan Muria 7 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

### 1.3.4 Membuat Relasi

Suatu tabel di dalam basis data dapat direlasikan atau dihubungkan dengan tabel yang lain. Misalkan tabel mahasiswa yang telah kita buat sebelumnya dapat direlasikan dengan dengan tabel skripsi yang memiliki struktur sebagai berikut:

Nama Atribut	Tipe Data	Ukuran Atribut	Keterangan
kode	varchar	5	PRIMARY KEY; kode skripsi
judul_skripsi	varchar	50	Judul skripsi
nim	varchar	12	Nomor induk mahasiswa

Untuk membuat relasi antara tabel mahasiswa dengan tabel skripsi ikuti langkah-langkah di bawah ini:

1. Membuat tabel baru dengan nama skripsi dimana struktur tabel sesuai dengan Tabel di atas.

```
mysql> CREATE TABLE skripsi(
  -> kode varchar(5),
  -> judul_skripsi varchar(50),
  -> nim varchar(12),
  -> PRIMARY KEY(kode)
  -> );
Query OK, 0 rows affected (0.04 sec)
```

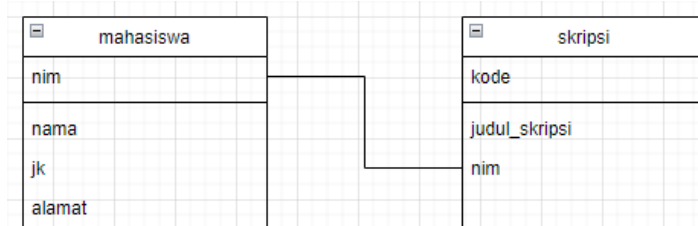
2. Melihat struktur tabel skripsi dengan perintah **DESCRIBE skripsi**.

```
mysql> DESCRIBE skripsi;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| kode       | varchar(5) | NO   | PRI | NULL    |       |
| judul_skripsi | varchar(50) | YES  |     | NULL    |       |
| nim        | varchar(12) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.03 sec)
```

3. Melihat daftar nama tabel yang ada di dalam basis data.

```
mysql> SHOW TABLES;
+-----+
| Tables_in_universitas |
+-----+
| mahasiswa              |
| skripsi                 |
+-----+
2 rows in set (0.01 sec)
```

4. Relasi yang terjadi diantara tabel mahasiswa dengan skripsi dapat digambarkan pada sebuah diagram relasi seperti yang ditunjukkan oleh gambar di bawah ini.



### 1.5 TUGAS PRAKTIKUM

1. Buatlah tabel baru untuk dosen, mata kuliah dan ambil\_mk sesuai dengan struktur tabel di bawah ini.

a. Tabel dosen

Nama Atribut	Tipe Data	Ukuran Atribut	Keterangan
NIP	varchar	18	PRIMARY KEY; kode dosen
nama_dosen	varchar	20	Nama dosen
alamat	varchar	20	Alamat dosen
no_hp	varchar	12	Nomor <i>handphone</i>

b. Tabel mata kuliah

Nama Atribut	Tipe Data	Ukuran Atribut	Keterangan
kode_mk	varchar	12	PRIMARY KEY; kode mata kuliah
nama_mk	varchar	20	Nama mata kuliah
sks	int	1	Sks mata kuliah

c. Tabel ambil\_mk

Nama Atribut	Tipe Data	Ukuran Atribut	Keterangan
NIM	varchar	12	Nomor induk mahasiswa
NIP	varchar	18	Kode dosen
kode_mk	varchar	12	Kode mata kuliah

- Gambarkan relasi yang terjadi di dalam keempat tabel yang sudah dibuat pada sebuah diagram relasi tabel seperti contoh sebelumnya.
- Lengkapi makna atau pengertian dari setiap jenis *relational key* yang ada di dalam basis data pada tabel di bawah ini!

No	Jenis Relasi	Pengertian
1	<i>Super Key</i>	
2	<i>Candidate Key</i>	
3	<i>Primary Key</i>	
4	<i>Alternate Key</i>	
5	<i>Foreign Key</i>	

- Berdasarkan relasi di antara tabel mahasiswa, dosen, mata kuliah, dan ambil\_mk identifikasi jenis relasi yang terdapat di dalamnya.
- Buatlah laporan praktikum untuk mendokumentasikan langkah-langkah praktikum yang telah dilakukan. Sertakan gambar dan penjelasannya dari masing-masing langkah.



## MODUL 3

### Pengantar SQL

#### 3.1 Bahasan dan Tujuan

##### 3.1.1 Bahasan

Modul 3 ini membahas tentang pengantar *Structured Query Language* (SQL) dan *Data Definition Language* (DDL)

##### 3.1.2 Tujuan

1. Mahasiswa memahami pengertian SQL.
2. Mahasiswa memahami klasifikasi SQL
3. Mahasiswa memahami konsep DDL
4. Mahasiswa memahami perintah DDL untuk konfigurasi database

#### 3.2 Dasar Teori

##### 3.2.1 Struktur Query Language

*Structured Query Language*(SQL) adalah serangkaian pernyataan pada engine database (termasuk engine Jet) yang berisi informasi apa yang ingin ditampilkan oleh pemakai. Kemudian engine memproses pernyataan tersebut dan menyediakan informasi yang diperlukan. SQL bukanlah bahasa pemrograman tetapi sub-language (subbahasa) yang berisi sekitar 30 pernyataan khusus dengan tugas mengelola database. Pernyataan SQL diintegrasikan pada bahasa pemrograman yang sebenarnya seperti visual basic.

Pernyataan SQL dikelompokkan menjadi dua yaitu DDL (*Data Definition Language*) dan DML (*Data Manipulation Language*). Pernyataan DDL dapat digunakan untuk membuat table, indeks, dan relasi database. Sedangkan pernyataan DML digunakan untuk memilih, mengurutkan, dan melakukan perhitungan terhadap data. Terdapat ketentuan dalam penulisan SQL, berikut aturan dalam penulisan SQL:

1. Semua keyword (kata kunci) dari pernyataan SQL diketik menggunakan huruf besar.
2. Informasi bertipe string yang terletak diantara pernyataan SQL dapat diapit dengankutip ganda (") atau kutip tunggal (').

3. Pada waktu menampilkan data (recordset), SQL mendukung menggunakan wildcards (memilih semua kolom atau field) dengan lambing asterisk (\*).
4. Jika nama field atau table memiliki spasi ditengahnya, maka nama tersebut harus diapit dengan brackets ( [ ] ). Contoh field dengan nama Data Pegawai dalam pernyataan SQL : [ Data Pegawai ]
5. Untuk menunjuk field khusus pada table khusus dalam pernyataan SQL digunakan notasi dot ( . ) NamaTabel>NamaField

Komponen SQL, *Data Definition Language* (DDL) : Digunakan untuk mendefinisikan data dengan menggunakan perintah : create, drop, alter.

#### **Command SQL:**

- CREATE :Membuat table, field, atau indeks.
- DROP Men-drop table atau indeks.
- ALTER Mengubah table dengan menambah field atau mengubah definisi field.

#### **Sedangkan untuk query dapat menggunakan klausa berikut ini:**

- FROM: Menentukan table mana yang datanya akan ditampilkan.
- WHERE Menentukan kondisi query.
- GROUP BY Menentukan group/kelompok dari informasi yang dipilih.
- HAVING Digunakan bersama.
- GROUP BY untk menentukan kondisi untuk tiap group dalam query.
- ORDER BY Menentukan urutan dari query.

### **3.2.2 Data Definition Language (DDL)**

#### **1. CREATE TABLE Fungsi**

: membuat tabel**Sintaks :**

CREATE TABLE tname(col 1

data type data spec,

col 2 data type data spec,

.

.

PRIMARY KEY (col1,.....));

**Contoh :**

```
CREATE TABLE PERSONEL (NOREG  
CHAR(10) NOT NULL,NAMA  
CHAR(45) NOT NULL, ALAMAT  
CHAR(45),  
TANGGAL_LAHIR NOT NULL WITH DEFAULT,PRIMARY KEY  
(NOREG));
```

**JENIS NULL**

Spesifikasi NULL, NOT NULL, NOT NULL WITH DEFAULT

- NULL :

Dapat diinterpretasikan sebagai nilai yang tidak diketahui atau tidak tersedianya suatu nilai.Null bukan berarti kosong (blank) atau 0 (No)

- NOT NULL :

pemakai atau program harus memberikan nilai-nilai pada saat memasukkan record

- NOT NULL WITH DEFAULT :

nilai default disimpan pada saat record dimasukkan tanpa nilai yang ditentukan untuk kolom ini.

Nilai default-nya :

No1 untuk tipe field NUMERIC Blank

untuk tipe field CHARACTER

CURRENT DATE untuk tipe field DATE

CURRENT TIME untuk tipe field TIME

Pada saat membuat tabel, salah satu atribut tersebut di atas dispesifikasikan pada sebuahkolom.

## 2. CREATE INDEX

Fungsi : membuat index

**Sintaks :**

```
CREATE [UNIQUE] INDEX indexnameON nama_table (nama_kolom);
```

**Contoh :**

```
CREATE UNIQUE INDEX KARYAWANHRD ON PERSONEL(NOREG);
```

Dengan indeks memungkinkan suatu tabel diakses dengan urutan tertentu tanpa harus merubah urutan fisik dari datanya dan dapat pula diakses secara cepat melalui indeks yang dibuat berdasar nilai field tertentu. Spesifikasi UNIQUE akan menolak key yang sama dalam file.

## 3. DROP TABLE

Fungsi : menghapus Tabel

**Sintaks :**

```
DROP TABLE tname;
```

**Contoh:**

```
DROP TABLE PERSONEL;
```

Dengan perintah itu obyek lain yang berhubungan dengan tabel tersebut otomatis akandihapus atau tidak akan berfungsi seperti :

- semua record dalam tabel akan terhapus
- index dan view pada tabel akan hilang
- deskripsi tabel akan hilang

## 4. DROP INDEX

Fungsi : menghapus index

**Sintaks :**

```
DROP INDEX indexname ;
```

**Contoh:**

```
DROP INDEX PRSONIDX;
```

## 5. ALTER

Fungsi : merubah atribut pada suatu tabel

### Sintaks :

ALTER TABLE tname

MODIFY nama\_kolom tipe\_kolom

ADD (nama\_kolom tipe\_kolom [[before, nama\_kolom]])

DROP (nama\_kolom tipe\_kolom)

### Contoh :

merubah Tabel JOBDESK dengan menambah Field LEMBUR.

```
ALTER TABLE JOBDESK ADD (LEMBUR CHAR(3));
```

### 3.3 TUGAS

#### 1. Membuat Database

```
CREATE DATABASE toko_modula;
```

#### 2. Membuat Tabel (CREATE TABLE)

```
a. CREATE TABLE Barang (  
    barang_ID varchar(5),  
    barang_nama varchar(20) NOT NULL,  
    pemasok varchar(5) NOT NULL,  
    barang_satuan varchar(15) NOT NULL,  
    barang_stok int NOT NULL,  
    harga int NOT NULL,  
    PRIMARY KEY(barang_ID),  
    FOREIGN KEY(pemasok) REFERENCES Pemasok(pemasok_ID)  
);
```

Input minimal 5 produk laptop dengan salah satu laptop harganya diatas 10 juta rupiah

```
b. CREATE TABLE Pemasok(  
    pemasok_ID varchar(5),  
    pemasok_nama varchar(20) NOT NULL,  
    pemasok_alamat varchar(6) NOT NULL,  
    pemasok_telp varchar(15) NOT NULL,  
    pemasok_email varchar(15) NOT NULL,  
    PRIMARY KEY(pemasok_ID)  
);
```

Input minimal 5 pemasok

```
c. CREATE TABLE Pegawai(  
    pegawai_ID varchar(5),  
    pegawai_nama varchar(20) NOT NULL,  
    pegawai_alamat varchar(20) NOT NULL,  
    pegawai_email varchar(15) NOT NULL,  
    PRIMARY KEY(pegawai_ID)  
);
```

d. Membuat Index

```
CREATE UNIQUE INDEX pemasok_idx ON Pemasok(pemasok_email);  
CREATE INDEX pemasok_idx1 ON Pemasok(pemasok_nama);  
CREATE INDEX barang_idx ON Barang(barang_stok);
```

**3. Modifikasi Tabel Barang dengan perintah :**

```
ALTER TABLE Barang ADD (barang_jenis varchar(20) NOT NULL);
```

Tunjukkan deskripsi tabel yang sudah dimodifikasi dengan menggunakan keyword DESC.

**4. Modifikasi Tabel Pegawai dengan perintah:**

a. ALTER TABLE Pegawai ADD (pegawai\_telp varchar(8) NOT NULL);

b. ALTER TABLE Pegawai MODIFY pegawai\_telp varchar(12) NOT NULL;

Tunjukkan deskripsi tabel menggunakan keyword DESC dan inputkan 5 data pegawai dengan nomor telepon sepanjang 12 digit.

Buatlah laporan praktikum untuk mendokumentasikan langkah-langkah praktikum yang telah dilakukan. Sertakan gambar dari masing-masing langkah. Kumpulkan laporan dalam format PDF dengan penamaan file:

<nama>\_<NIM>\_<kelas dalam huruf>\_Praktikum 3.pdf

Contoh: Ananda\_20040567001\_A\_Praktikum3.pdf

## MODUL 4

### DATA MANAGEMENT LANGUAGE (DML)

#### 4.1 Bahasan dan Tujuan

##### 4.1.1 Bahasan

Membahas tentang perintah-perintah Data Management Language (DML) untuk menampilkan, menambah, mengubah, dan menghapus data pada tabel.

##### 4.1.2 Tujuan

1. Mahasiswa memahami perintah SQL untuk menginputkan data ke dalam tabel MySQL.
2. Mahasiswa memahami perintah SQL untuk mengubah data ke dalam tabel MySQL.
3. Mahasiswa memahami perintah SQL untuk menghapus data ke dalam tabel MySQL.
4. Mahasiswa memahami perintah SQL untuk menampilkan data ke dalam tabel MySQL

#### 4.2 Dasar Teori

##### 4.2.1 Menginputkan Data

Sebuah data dapat dimasukkan ke dalam tabel menggunakan perintah INSERT. Namun hal yang perlu diperhatikan yaitu struktur tabel yang akan digunakan untuk memasukkan data. Sebagai contoh, berikut ini terdapat Tabel "mata\_kuliah" dengan struktur sebagai berikut:

kode_mk	Varchar (12)
nama_mk	Varchar (20)
Sks	Integer

Untuk menambahkan data pada tabel, kita menggunakan perintah:

```
INSERT INTO mata_kuliah VALUES ("122000768112","Praktikum Basis Data",2);
```

Berikut penjelasannya:

1. Field kode\_mk dan nama\_mk bertipe Varchar sehingga data diinputkan dalam tanda petik ("").
2. Field sks bertipe Integer sehingga data diinputkan tanpa tanda petik ("").



#### 4.2.2 Menampilkan Data

Dalam database, perintah SQL untuk menampilkan data sebuah tabel bisa menggunakan SELECT. Berikut struktur SQL untuk penampilan data berdasarkan kolom tertentu :

```
SELECT nama_kolom FROM nama_tabel;
```

Jika ingin menampilkan keseluruhan kolom dan keseluruhan baris suatu tabel bisa mengganti nama\_kolom menggunakan tanda asterisk (\*). Kemudian, bila ingin menampilkan berdasarkan baris tertentu bisa menggunakan perintah WHERE yang diletakkan dibelakang nama tabel dan isikan nama kolom beserta kata kunci sebagai dasar atau syarat dari pencarian baris. Untuk lebih jelasnya berikut struktur perintah SQL nya :

```
SELECT nama_kolom FROM nama_tabel WHERE nama_kolom = kata kunci;
```

Apabila ingin menampilkan data dari lebih dari satu tabel, struktur SQL nyaterdapat sedikit perbedaan. Misalnya memakai tabel yang telah dibuat pada Modul 2 yaitu mahasiswa dan skripsi (dengan syarat adanya relasi antara kedua tabel tersebut). Berikut Struktur SQL nya :

```
SELECT nama_kolom, nama_kolom, nama_kolom, nama_kolom FROM nama_tabel_1,  
nama_tabel_2 WHERE nama_tabel_1.nama_kolom = nama_tabel_2.nama_kolom;
```

Contoh:

```
SELECT mahasiswa.nama, skripsi.judul_skripsi FROM mahasiswa, skripsi WHERE mahasiswa.nim =  
skripsi.nim;
```

#### 4.2.3 Menghapus Data

Pada database PostgreSQL, DELETE digunakan untuk menghapus data pada sebuah tabel. Berikut Struktur yang digunakan :

```
DELETE FROM namatabel;
```

#### 4.2.4 Mengubah Data

Dalam memodifikasi database bisa menggunakan perintah UPDATE. Berikut contoh struktur SQL untuk memodifikasi data pada kolom tertentu berdasarkan baris tertentu:

**update namatabel set namakolom = isidata where namakolom = katakunci;**

#### 4.3 PRAKTIKUM

1. Pada praktikum ini, kita akan menggunakan basis data bernama “universitas” yang telah kita buat pada Modul 2 Objek basis Data. Aktifkan basis data dengan menuliskan sintaks SQL berikut ini:

```
mysql> USE universitas;  
Database changed  
mysql>
```

2. Inputkan data-data di bawah ini ke dalam Basis Data “universitas”!

a. Tabel Data Mahasiswa (mahasiswa)

NIM	Nama	Jenis Kelamin	Alamat
138572311001	Nanda	P	Sidoarjo
138572311002	Dipa	L	Blitar
138624411001	Cantika	P	Blitar
138624411002	Aditya	P	Tulungagung
138752411001	Krisna	P	Surabaya

b. Tabel Data Skripsi (skripsi)

Kode Skripsi	Judul Skripsi	NIM
16001	Sistem Penjadwalan Perkuliahan dengan Metode ABC	138572311001
16002	Sistem Pendukung Keputusan Pemilihan Pegawai Terbaik di PT. Cahaya	138572311002
16003	Peramalan Tingkat Inflasi di Kota Blitar dengan Metode XYZ	138624411001
16004	Rancang Bangun Sistem Informasi Akademik di SMP Merah Putih	138624411002
16005	Rancang Bangun Sistem Informasi Kepegawaian di Kantor Pemerintah Kota Surabaya	138752411001

c. Tabel Data Dosen (dosen)

NIP	Nama	Alamat	No. HP
198503042002101005	Agus	Malang	085467788
198305222013112002	Ratih	Malang	085474423
198608202009091003	Rudi	Sidoarjo	084678769

198707262011072001	Putri	Sidoarjo	085556777
198407042016122001	Fakhriyah	Jember	0824563672

d. Tabel Data Mata Kuliah (mata\_kuliah)

Kode MK	Nama MK	SKS
200500123001	Basis Data	3
200500123002	Algoritma dan Pemrograman	3
200500123003	Pengantar Ilmu Komputer	3
200500123004	Pancasila	2
200500123005	Bahasa Indonesia	2

e. Tabel Data Pengambilan Mata Kuliah (ambil\_mk)

NIM	NIP	Kode MK
138572311001	198305222013112002	200500123001
138572311002	198608202009091003	200500123002
138624411001	198707262011072001	200500123002
138572311001	198407042016122001	200500123004
138572311002	198407042016122001	200500123004

Tampilkan hasil input data pada masing-masing tabel!

2. Lakukan update data berikut pada Basis Data “universitas”:

- Ubah alamat mahasiswa dengan NIM 138572311001 menjadi “Tulungagung”. Tampilkan hasilnya!
- Ubah nama mata kuliah dengan Kode MK 200500123003 menjadi “Pengantar Teknologi Informasi”. Tampilkan hasilnya.
- Ubah judul skripsi untuk mahasiswa dengan NIM 138572311001 menjadi “Peramalan Penjualan untuk PT. Angkasa”. Tampilkan hasilnya!

3. Tampilkan data berikut ini dengan menggunakan query SQL:

- Data seluruh mahasiswa.
- Data mahasiswa yang tinggal di Blitar.
- Data mata kuliah yang memiliki 3 sks.
- Data nama mahasiswa yang mengambil Mata Kuliah Pancasila.
- Data alamat dosen yang mengampu Mata Kuliah Pancasila.

Buat laporan praktikum dalam format .pdf dan ikuti format penamaan file berikut:

<nama>\_<NIM>\_<kelas dalam huruf>\_Praktikum 1.pdf

Contoh: Ananda\_20040567001\_A\_Praktikum1.pdf

## MODUL 5

### SELEKSI DATA

#### 5.1 Bahasan dan Tujuan

##### 5.1.1 Bahasan

Membahas tentang perintah dan operator-operator yang digunakan untuk melakukan seleksi data dengan kriteria tertentu pada tabel.

##### 5.1.2 Tujuan

1. Mahasiswa mampu memahami perintah DISTINCT untuk menampilkan data dari tabel.
2. Mahasiswa mampu memahami perintah WHERE untuk menampilkan data dari tabel.
3. Mahasiswa mampu memahami operator-operator seleksi data SQL untuk menampilkan data dengan kriteria khusus dari tabel.
4. Mahasiswa mampu memahami perintah LIKE untuk menampilkan data dari tabel.

#### 5.2 Dasar Teori

##### 5.2.1 SELECT DISTINCT

Di dalam tabel sering kali terdapat data yang nilainya sama. Kadang kala kita menginginkan untuk menampilkan data yang berbeda dari suatu tabel. Perintah DISTINCT digunakan untuk menampilkan data yang nilainya berbeda pada tabel, sehingga nilai yang sama hanya akan ditampilkan satu kali. Perintah SQL untuk DISTINCT yaitu

```
SELECT DISTINCT nama_kolom FROM nama_tabel;
```

Misalkan kita menginginkan data alamat yang berbeda dari tabel Mahasiswa.

NIM	Nama	Jenis Kelamin	Alamat
138572311001	Nanda	P	Sidoarjo
138572311002	Dipa	L	Blitar
138624411001	Cantika	P	Blitar
138624411002	Aditya	P	Tulungagung
138752411001	Krisna	P	Surabaya

Perintah SQL yang digunakan adalah sebagai berikut:

```
SELECT DISTINCT alamat FROM mahasiswa;
```

```
+-----+  
| alamat |  
+-----+  
| Sidoarjo |  
| Blitar |  
| Tulungagung |  
| Surabaya |  
+-----+  
4 rows in set (0.02 sec)
```

### 5.2.2 Perintah WHERE

Perintah WHERE digunakan untuk menampilkan data pada tabel dengan kriteria atau syarat tertentu.

Struktur perintah SQL nya dapat dilihat pada perintah di bawah ini:

```
SELECT nama_kolom FROM nama_tabel WHERE nama_kolom = kata kunci;
```

Manfaatkan kembali tabel mahasiswa pada praktikum modul sebelumnya. Misalkan kita ingin menampilkan data mahasiswa yang berjenis kelamin pria. Perintah SQL yang digunakan adalah

```
SELECT * FROM mahasiswa WHERE jenis_kelamin = 'P';
```

```
+-----+-----+-----+-----+  
| nim      | nama      | jenis_kelamin | alamat |  
+-----+-----+-----+-----+  
| 138572311001 | Nanda     | P             | Sidoarjo |  
| 138624411001 | Cantika   | P             | Blitar   |  
| 138624411002 | Aditya    | P             | Tulungagung |  
| 138752411001 | Krisna    | P             | Surabaya |  
+-----+-----+-----+-----+  
4 rows in set (0.02 sec)
```

### 5.2.3 Operator Seleksi

#### a. Operator AND, OR, NOT

Dalam penggunaannya perintah WHERE dapat dikombinasikan dengan operator AND, OR dan NOT. Operator AND dan OR digunakan untuk melakukan menyeleksi data tabel dengan kondisi lebih dari 1.

Operator AND akan memberikan hasil seleksi apabila semua kondisi bernilai benar (TRUE).

Operator OR akan memberikan hasil seleksi apabila salah satu kondisi bernilai benar.

Perintah SQL untuk penggunaan operator AND yaitu:

```
SELECT nama_kolom1, nama_kolom2, ...  
FROM nama_tabel  
WHERE kondisi1 AND kondisi2 AND kondisi3 ...;
```

Sebagai contoh kita ingin menampilkan data mahasiswa yang berjenis kelamin pria dan berasal dari blitar. Perintah SQL:

```
SELECT * FROM mahasiswa WHERE jenis_kelamin='P' AND alamat="Blitar";
```

```
+-----+
| nim   | nama  | jenis_kelamin | alamat |
+-----+
| 138624411001 | Cantika | P             | Blitar |
+-----+
1 row in set (0.08 sec)
```

Sedangkan perintah SQL untuk penggunaan operator OR yaitu:

```
SELECT nama_kolom1, nama_kolom2, ...
FROM nama_tabel
WHERE kondisi1 OR kondisi2 OR kondisi3 ...;
```

Sebagai contoh kita ingin menampilkan data dosen yang alamatnya di malang atau sidoarjo.

Perintah SQLnya yaitu:

```
SELECT * FROM dosen WHERE alamat="Malang" OR alamat="Sidoarjo";
```

```
+-----+
| NIP      | Nama  | Alamat  | No_HP |
+-----+
| 198305222013112002 | Ratih | Malang  | 085474423 |
| 198503042002101005 | Agus  | Malang  | 085467788 |
| 198608202009091003 | Rudi  | Sidoarjo | 084678769 |
| 198707262011072001 | Putri | Sidoarjo | 085556777 |
+-----+
4 rows in set (0.02 sec)
```

Operator NOT akan menampilkan data yang nilainya berlawanan dengan kondisi yang diberikan (NOT TRUE). Perintah SQL untuk penggunaan operator NOT adalah

```
SELECT nama_kolom1, nama_kolom2, ...
FROM nama_tabel
WHERE NOT kondisi;
```

Misalkan kita ingin menampilkan nama mahasiswa yang bukan perempuan. Sehingga perintah SQL yang kita gunakan yaitu

```
SELECT nama FROM mahasiswa WHERE NOT jenis_kelamin='P';
```

```
+-----+
| nama |
+-----+
| Dipa |
+-----+
1 row in set (0.02 sec)
```

## b. Operator IN dan NOT IN

Operator IN memungkinkan kita untuk menentukan beberapa kondisi pada perintah WHERE. Dengan operator IN kita dapat mempersingkat penulisan kondisi pada operator OR. Perintah SQL

penggunaan dari operator IN yaitu

```
SELECT nama_kolom1, nama_kolom2,...  
FROM nama_tabel  
WHERE nama_kolom IN (nilai1, nilai2, ...);
```

Sebagai contoh kita ingin menampilkan data dosen yang alamatnya di malang atau sidoarjo.

Perintah SQL nya yaitu

```
SELECT * FROM dosen WHERE alamat IN ("Malang", "Sidoarjo");
```

```
+-----+-----+-----+-----+  
| NIP          | Nama  | Alamat  | No_HP    |  
+-----+-----+-----+-----+  
| 198305222013112002 | Ratih | Malang  | 085474423 |  
| 198503042002101005 | Agus  | Malang  | 085467788 |  
| 198608202009091003 | Rudi  | Sidoarjo | 084678769 |  
| 198707262011072001 | Putri | Sidoarjo | 085556777 |  
+-----+-----+-----+-----+  
4 rows in set (0.03 sec)
```

Sedangkan operator NOT IN digunakan untuk menampilkan data yang nilainya berlawanan dengan kondisi yang diberikan. Perintah NOT IN ini juga dapat mempersingkat dalam penulisan kondisi dengan operator NOT. Sebagai contoh kita ingin menampilkan data dosen yang alamatnya selain malang dan sidoarjo. Sehingga perintah SQL nya yaitu

```
SELECT * FROM dosen WHERE alamat NOT IN ("Malang", "Sidoarjo");
```

```
+-----+-----+-----+-----+  
| NIP          | Nama      | Alamat  | No_HP    |  
+-----+-----+-----+-----+  
| 198407042016122001 | Fakhriyah | Jember  | 0824563672 |  
+-----+-----+-----+-----+  
1 row in set (0.02 sec)
```

### c. Operator BETWEEN

Operator BETWEEN akan menampilkan data yang nilainya diantara nilai kondisi yang diberikan. Nilai dari kondisi operator BETWEEN bisa berupa nilai angka, teks, atau tanggal. Sebagai contoh kita ingin menampilkan data mata kuliah yang sksnya diantara 1 dan 3. Sehingga perintah SQL yang kita gunakan yaitu

```
SELECT * FROM mata_kuliah WHERE sks BETWEEN 1 AND 3;
```

```
+-----+-----+-----+  
| kode_mk      | nama_mk          | sks |  
+-----+-----+-----+  
| 200500123001 | Basis Data       | 3   |  
| 200500123002 | Algoritma dan Pemrograman | 3   |  
| 200500123003 | Pengantar Ilmu Komputer | 3   |  
| 200500123004 | Pancasila        | 2   |  
| 200500123005 | Bahasa Indonesia | 2   |  
| 200500123006 | Praktikum Basis Data | 1   |  
| 200500123007 | Praktikum Algoritma dan Pemrograman | 1   |  
+-----+-----+-----+  
7 rows in set (0.00 sec)
```

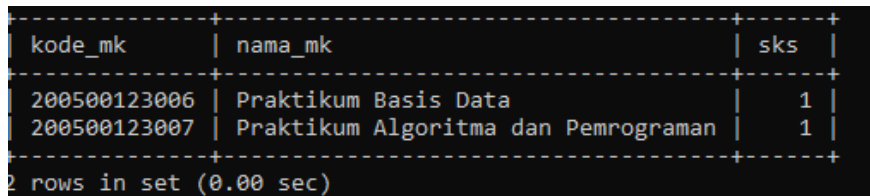
#### d. Operator LIKE

Operator LIKE memungkinkan pengambilan data yang mirip atau mendekati kata kunci yang diberikan. Berikut contoh penggunaan dari operator LIKE:

Kita ingin menampilkan data mata kuliah yang nama mata kuliahnya berawalan "Praktikum".

Perintah SQL:

```
SELECT * FROM mata_kuliah WHERE nama_mk LIKE "Praktikum%";
```



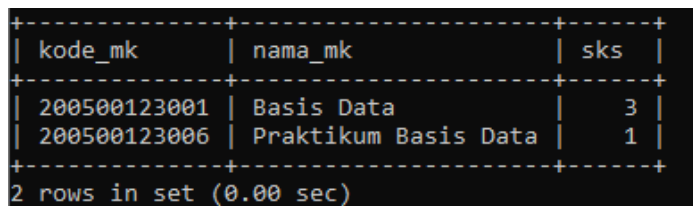
kode_mk	nama_mk	sks
200500123006	Praktikum Basis Data	1
200500123007	Praktikum Algoritma dan Pemrograman	1

2 rows in set (0.00 sec)

Selanjutnya kita ingin menampilkan data mata kuliah yang nama mata kuliahnya berakhiran "Data".

Perintah SQL:

```
SELECT * FROM mata_kuliah WHERE nama_mk LIKE "%Data";
```



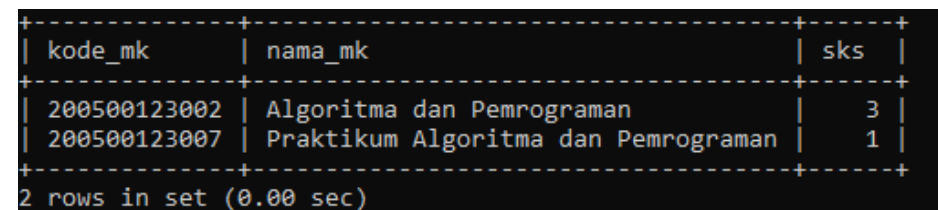
kode_mk	nama_mk	sks
200500123001	Basis Data	3
200500123006	Praktikum Basis Data	1

2 rows in set (0.00 sec)

Contoh yang terakhir adalah kita ingin menampilkan data mata kuliah yang nama mata kuliahnya mengandung kata "Algo".

Perintah SQL:

```
SELECT * FROM mata_kuliah WHERE nama_mk LIKE "%Algo%";
```



kode_mk	nama_mk	sks
200500123002	Algoritma dan Pemrograman	3
200500123007	Praktikum Algoritma dan Pemrograman	1

2 rows in set (0.00 sec)

#### 5.2.4 Pengurutan Data

Mengurutkan data di dalam basis data dapat dilakukan dengan menggunakan perintah ORDER BY. Di mana nilai *default* dari perintah ORDER BY yaitu ASC (*ascending* atau pengurutan naik). Contoh dari penggunaan ORDER BY sebagai berikut:

Kita ingin menampilkan data mata kuliah berdasarkan besaran nilai sksnya dengan urutan menaik (*ascending*). Perintah SQL:

```
SELECT * FROM mata_kuliah ORDER BY sks;
```



```
+-----+-----+-----+
| kode_mk | nama_mk | sks |
+-----+-----+-----+
| 200500123006 | Praktikum Basis Data | 1 |
| 200500123007 | Praktikum Algoritma dan Pemrograman | 1 |
| 200500123004 | Pancasila | 2 |
| 200500123005 | Bahasa Indonesia | 2 |
| 200500123001 | Basis Data | 3 |
| 200500123002 | Algoritma dan Pemrograman | 3 |
| 200500123003 | Pengantar Ilmu Komputer | 3 |
| 200500123008 | Skripsi | 6 |
+-----+-----+-----+
8 rows in set (0.03 sec)
```

Selanjutnya kita ingin menampilkan data mata kuliah yang diurutkan berdasarkan nama mata kuliah dengan urutan menurun (*descending*). Perintah SQL:

```
SELECT * FROM mata_kuliah ORDER BY nama_mk DESC;
```

```
+-----+-----+-----+
| kode_mk | nama_mk | sks |
+-----+-----+-----+
| 200500123008 | Skripsi | 6 |
| 200500123006 | Praktikum Basis Data | 1 |
| 200500123007 | Praktikum Algoritma dan Pemrograman | 1 |
| 200500123003 | Pengantar Ilmu Komputer | 3 |
| 200500123004 | Pancasila | 2 |
| 200500123001 | Basis Data | 3 |
| 200500123005 | Bahasa Indonesia | 2 |
| 200500123002 | Algoritma dan Pemrograman | 3 |
+-----+-----+-----+
8 rows in set (0.02 sec)
```

### 5.3 TUGAS PRAKTIKUM

1. Tampilkan data nim dan nama mahasiswa yang namanya mengandung huruf 'i' dan berjenis kelamin perempuan.
2. Tampilkan data mata kuliah yang namanya tidak mengandung "Basis Data" dan sksnya lebih dari 2.
3. Tampilkan data dosen yang diurutkan berdasarkan nama dengan urutan menurun.
4. Tampilkan judul skripsi yang diawali dengan kata sistem atau mengandung kata metode.
5. Lakukan penambahan atribut tanggal lahir pada tabel dosen dengan tipe data date. Selanjutnya lakukan pengisian data pada atribut tanggal lahir seperti pada tabel di bawah ini.

```
+-----+-----+-----+-----+-----+
| NIP | Nama | Alamat | No_HP | tanggal_lahir |
+-----+-----+-----+-----+-----+
| 198305222013112002 | Ratih | Malang | 085474423 | 1983-05-22 |
| 198407042016122001 | Fakhriyah | Jember | 0824563672 | 1984-07-04 |
| 198503042002101005 | Agus | Malang | 085467788 | 1985-03-04 |
| 198608202009091003 | Rudi | Sidoarjo | 084678769 | 1986-08-20 |
| 198707262011072001 | Putri | Sidoarjo | 085556777 | 1986-07-26 |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Tampilkan data dosen yang tahun lahirnya diantara tahun 1985 dan 1986 order by *ascending*.

## MODUL 6

### FUNGSI AGREGAT

#### 6.1 Bahasan dan Tujuan

##### 6.1.1 Bahasan

Fungsi Agregat untuk menghitung beberapa nilai dan mengembalikan hasilnya sebagai nilai tunggal, seperti rata-rata semua nilai, jumlah semua nilai, dan nilai maksimum dan minimum di antara pengelompokan nilai.

##### 6.1.2 Tujuan

1. Memahami fungsi-fungsi agregat dan penggunaannya.
2. Memahami operasi pengelompokan data.
3. Mampu menyelesaikan kasus-kasus yang melibatkan penggunaan fungsi- fungsi agregat.
4. Mampu menyelesaikan kasus-kasus yang melibatkan penggunaan fungsi- fungsi agregat dan pengelompokan.

#### 6.2 Dasar Teori

##### 6.2.1. Fungsi Agregat

Fungsi Agregat (aggregate) adalah fungsi yang menerima koleksi nilai dan mengembalikan nilai tunggal sebagai hasilnya. Standar ISO mendefinisikan lima jenis fungsi agregat.

Fungsi	Deskripsi
COUNT	Mengembalikan jumlah (banyaknya atau kemunculannya) nilai di suatu kolom
SUM	Mengembalikan jumlah (total atau sum) nilai di suatu kolom
AVG	Mengembalikan rata-rata (average) nilai di suatu kolom
MIN	Mengembalikan nilai terkecil (minimal) di suatu kolom
MAX	Mengembalikan nilai terbesar (maximal) di suatu kolom

Buatlah database “Fungsi\_Agregat”, kemudian buatlah tabel “Matkul” berikut:

Kode_mk	Nama_mk	sks	semester
MKP1011	Calculus	3	3
MKP1213	Kewarganegaraan	2	5
MKP6789	Bahasa Arab I	1	3
MKU1415	Teosofi	3	5
MKU2345	Pancasila	1	3
MKW1617	Digital Electronic	3	5
MKW1819	Bahasa Inggris I	2	5

### 6.2.2. Select DISTINCT

Select DISTINCT dapat dimanfaatkan untuk mengeliminasi duplikasi kemunculan data yang sama.

Sintaks Select DISTINCT sebagai berikut:

```
SELECT DISTINCT nama_kolom FROM nama_tabel;
```

Contoh:

Menampilkan kolom semester dari tabel Matkul.

```
SELECT DISTINCT semester FROM Matkul;
```

```
+-----+
| semester |
+-----+
| 3        |
| 5        |
+-----+
```

### 6.2.3. Pengelompokan

GROUP BY digunakan untuk mengelompokkan data (record) yang memiliki nilai yang sama. seperti “menemukan jumlah data barang sesuai dengan kategori”. SQL GROUP BY sering digunakan pada fungsi agregat seperti (COUNT(), MAX(), MIN(), SUM(), AVG()) yang

menampilkan beberapa kolom.

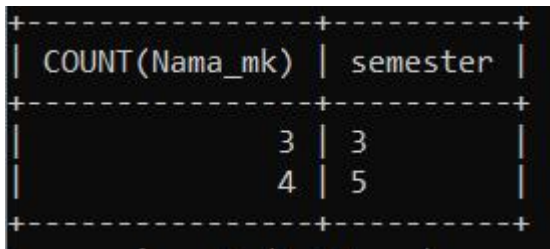
Sintaks Select GROUP BY sebagai berikut:

```
SELECT nama_kolom FROM nama_tabel [WHERE kondisi] GROUP BY nama_kolom  
ORDER BY nama_kolom;
```

Contoh:

Menampilkan jumlah mata kuliah, dikelompokkan berdasarkan semester.

```
SELECT COUNT>Nama_mk), semester FROM Matkul GROUP BY semester;
```



COUNT>Nama_mk)	semester
3	3
4	5

### 6.2.3. Having

Pada saat bekerja dengan fungsi agregat, terkadang diperlukan klausa WHERE untuk menspesifikasikan hasil. Dikarenakan, klausa WHERE tidak boleh mengandung fungsi agregat. Sebagai solusi, dapat menggunakan klausa HAVING. Penggunaan klausa ini mirip WHERE.

Sintaks Select HAVING sebagai berikut:

```
SELECT nama_kolom FROM nama_tabel [WHERE kondisi] GROUP BY nama_kolom  
HAVING kondisi ORDER BY nama_kolom;
```

Contoh:

Menampilkan nama mata kuliah dan semester berdasarkan kode mk dengan ketentuan, yang dapat diambil sebelum semester 5.

```
SELECT Nama_mk, SUM(semester) FROM Matkul GROUP BY Kode_mk HAVING  
SUM(semester) <5;
```

Nama_mk	SUM(semester)
Calculus	3
Bahasa Arab I	3
Pancasila	3

#### 6.2.4 Fungsi COUNT

COUNT digunakan untuk menghitung jumlah record.

Sintaks dasar COUNT sebagai berikut:

```
SELECT COUNT(*) FROM nama_tabel;
```

Sintaks menggunakan nama alias :

```
SELECT COUNT(*) AS nama_alias_kolom FROM nama_table;
```

Contoh:

```
SELECT COUNT(*) FROM Matkul;
```

COUNT(*)
7

```
SELECT COUNT(*) AS jumlah_data_masuk FROM Matkul;
```

jumlah_data_masuk
7

#### 6.2.5 Fungsi SUM

SUM digunakan untuk menghitung total nilai dari kolom tertentu.

Sintaks dasar SUM sebagai berikut:

```
SELECT SUM(nama_kolom) FROM nama_tabel;
```

Sintaks menggunakan nama alias :

```
SELECT SUM(nama_kolom) AS nama_alias_kolom FROM nama_tabel;
```

Contoh:

```
SELECT SUM(sks) FROM Matkul;
```

```
+-----+  
| SUM(sks) |  
+-----+  
|      15 |  
+-----+
```

```
SELECT SUM(sks) AS total_sks FROM Matkul;
```

```
+-----+  
| total_sks |  
+-----+  
|      15 |  
+-----+
```

### 6.2.6 Fungsi AVG

AVG digunakan untuk menghitung total nilai dari kolom tertentu.

Sintaks dasar AVG sebagai berikut:

```
SELECT AVG(nama_kolom) FROM nama_table;
```

Sintaks menggunakan nama alias :

```
SELECT AVG(nama_kolom) AS nama_alias_kolom FROM nama_table;
```

Contoh:

```
SELECT AVG(sks) FROM Matkul;
```

```
+-----+  
| AVG(sks) |  
+-----+  
| 2.142857142857143 |  
+-----+
```

```
SELECT AVG(sks) AS rata_rata_sks FROM Matkul;
```

```
+-----+
| rata_rata_sks |
+-----+
| 2.142857142857143 |
+-----+
```

### 6.2.7 Fungsi MIN

MIN digunakan untuk menampilkan nilai terendah dari suatu kolom.

Sintaks dasar MIN sebagai berikut:

```
SELECT MIN(nama_kolom) FROM nama_table;
```

Sintaks menggunakan nama alias :

```
SELECT MIN(nama_kolom) AS nama_alias_kolom FROM nama_table;
```

Contoh:

```
SELECT MIN(sks) FROM Matkul;
```

```
+-----+
| MIN(sks) |
+-----+
| 1        |
+-----+
```

```
SELECT MIN(sks) AS minimal_sks FROM Matkul;
```

```
+-----+
| minimal_sks |
+-----+
| 1          |
+-----+
```

### 6.2.8 Fungsi MAX

MAX digunakan untuk menampilkan nilai tertinggi dari suatu kolom.

Sintaks dasar MAX sebagai berikut:

```
SELECT MAX(nama_kolom) FROM nama_table;
```

Sintaks menggunakan nama alias :

```
SELECT MAX(nama_kolom) AS nama_alias_kolom FROM nama_table;
```

Contoh:

```
SELECT MAX(sks) FROM Matkul;
```

```
+-----+
| MAX(sks) |
+-----+
| 3         |
+-----+
```

```
SELECT MAX(sks) AS maksimal_sks FROM Matkul;
```

```
+-----+
| maksimal_sks |
+-----+
| 3            |
+-----+
```

TUGAS!

1. Buatlah database “kantor\_agregrat”, kemudian buatlah tabel “jam\_kerja” berikut:

Kode_karyawan	Nama_kywn	PSW	total_jam_kerja	Gaji
50001	Anggi	1	7	4500000
50002	Benny	1	6	4000000
50003	Cherill	0	7	5000000
50004	Davi	3	5	3500000
50005	Ega	1	6	4000000
50006	Fachri	2	5	3500000
50007	Gaga	5	2	3500000

2. Ubah nama Cherill menjadi “nama lengkap masing- masing”. Tampilkan hasilnya beserta sintaks!
3. Tampilkan jumlah gaji karyawan, pada tabel ”jam\_kerja”, berikan nama inisial kolom “jumlah gaji karyawan” serta tampilkan hasilnya beserta sintaks!
4. Tampilkan kode karyawan, nama karyawan, dan gaji, dikelompokkan berdasarkan kode karyawan dengan gaji  $\geq 4000000$ , serta tampilkan hasilnya beserta sintaks!



Buatlah laporan praktikum untuk mendokumentasikan langkah-langkah praktikum mulai dari pembuatan database “kantor\_agregat” yang telah dilakukan hingga tugas. Sertakan gambar dari masing-masing langkah, beserta penjelasan singkat.

Kumpulkan laporan dalam format PDF dengan penamaan file:

<nama>\_<NIM>\_<kelas dalam huruf>\_Praktikum6.pdf

Contoh: Ananda\_20040567001\_A\_Praktikum6.pdf

## MODUL 7

### OPERATOR JOIN

#### 7.1 Bahasan dan Tujuan

##### 7.1.1 Bahasan

Membahas tentang Operator Join pada MySQL, meliputi Inner Join, Left Outer Join, Right Outer Join, dan Cross Join.

##### 7.1.2 Tujuan

Mahasiswa memahami perintah SQL untuk menginputkan melakukan Inner Join, Left Outer Join, Right Outer Join, dan Cross Join.

#### 7.2 Dasar Teori

Secara umum, Operator JOIN pada SQL digunakan untuk **menggabungkan data dari 2 atau lebih tabel pada basis data**. Pada SQL, dikenal beberapa Operator JOIN, yaitu INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, dan CROSS JOIN.

##### 7.2.1 Mengenal Perintah INNER JOIN

Operator INNER JOIN digunakan untuk menggabungkan data dari dua tabel dengan hanya menampilkan irisan dari kedua tabel tersebut. Sebagai contoh, terdapat 2 tabel bernama **ambil\_mk** dan **mata\_kuliah**, seperti di bawah ini:

##### 1. Tabel ambil\_mk

```
mysql> SELECT * FROM ambil_mk;
+-----+-----+-----+
| nim      | nip      | kode_mk |
+-----+-----+-----+
| 138572311001 | 198305222015112002 | 200500123001 |
| 138572311002 | 198608202009091003 | 200500123002 |
| 138624411001 | 198707262011072001 | 200500123002 |
| 138572311001 | 198407042016122001 | 200500123004 |
| 138572311002 | 198407042016122001 | 200500123004 |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

## 2. Tabel mata\_kuliah

```
mysql> SELECT * FROM mata_kuliah;
+-----+-----+-----+
| kode_mk | nama_mk | sks |
+-----+-----+-----+
| 200500123001 | Basis Data | 3 |
| 200500123002 | Algoritma dan Pemrograman | 3 |
| 200500123003 | Pengantar Ilmu Komputer | 3 |
| 200500123004 | Pancasila | 3 |
| 200500123005 | Bahasa Indonesia | 3 |
| 200500123006 | Praktikum Basis Data | 1 |
| 200500123007 | Praktikum Algoritma dan Pemrograman | 1 |
+-----+-----+-----+
7 rows in set (0.00 sec)
```

Untuk menampilkan NIM mahasiswa dan nama mata kuliah yang diambil, kita perlu melakukan penggabungan terhadap kedua tabel di atas dengan menggunakan **INNER JOIN**.

**Contoh:**

```
SELECT a.nim, b.nama_mk
FROM ambil_mk a
INNER JOIN mata_kuliah b
ON a.kode_mk = b.kode_mk;
```

Pada query di atas, klausa **ON** diikuti oleh kolom yang merupakan foreign key. Pada contoh di atas, kedua tabel memiliki sebuah kolom yang merupakan foreign key, yaitu kode\_mk. Karena foreign key tersebut memiliki nama yang sama pada kedua tabel (yaitu kode\_mk), ini disebut dengan **natural join**.

Hasil dari query di atas:

```
mysql> select a.nim, b.nama_mk
-> FROM ambil_mk a INNER JOIN mata_kuliah b
-> ON a.kode_mk=b.kode_mk;
+-----+-----+
| nim | nama_mk |
+-----+-----+
| 138572311001 | Basis Data |
| 138572311002 | Algoritma dan Pemrograman |
| 138624411001 | Algoritma dan Pemrograman |
| 138572311001 | Pancasila |
| 138572311002 | Pancasila |
+-----+-----+
5 rows in set (0.00 sec)
```

Kita juga dapat menggunakan **Operator INNER JOIN** pada **3 tabel**, contohnya untuk menampilkan nama dosen dan mata kuliah yang diampunya dan sudah diambil, yaitu:

```
SELECT b.nama, c.nama_mk
FROM ambil_mk a INNER JOIN dosen b ON a.nip=b.nip
INNER JOIN mata_kuliah c ON a.kode_mk = c.kode_mk;
```

Hasil dari query di atas:

```
mysql> SELECT b.nama, c.nama_mk
-> FROM ambil_mk a INNER JOIN dosen b ON a.nip=b.nip
-> INNER JOIN mata_kuliah c ON a.kode_mk=c.kode_mk;
+-----+-----+
| nama   | nama_mk |
+-----+-----+
| Ratih  | Basis Data |
| Rudi   | Algoritma dan Pemrograman |
| Putri  | Algoritma dan Pemrograman |
| Fakhriyah | Pancasila |
| Fakhriyah | Pancasila |
+-----+-----+
5 rows in set (0.00 sec)
```

## 7.2.2 Mengetahui Operator RIGHT OUTER JOIN

Berbeda dengan Operator INNER JOIN, Operator RIGHT OUTER JOIN digunakan untuk menggabungkan dua tabel dengan menampilkan seluruh data pada tabel yang ada di **sebelah kanan Operator RIGHT OUTER JOIN**. Sebagai contoh, kita akan menggabungkan data dari dua tabel pada bagian sebelumnya untuk menampilkan data nim dan nama mata\_kuliah dengan **menampilkan seluruh data pada Tabel mata\_kuliah**.

Contoh:

```
SELECT a.nim, b.nama_mk
FROM ambil_mk a RIGHT OUTER JOIN mata_kuliah b
ON a.kode_mk = b.kode_mk;
```

Hasil dari query di atas:

```
mysql> SELECT a.nim, b.nama_mk
-> FROM ambil_mk a RIGHT OUTER JOIN mata_kuliah b
-> ON a.kode_mk = b.kode_mk;
+-----+-----+
| nim      | nama_mk |
+-----+-----+
| 138572311001 | Basis Data |
| 138624411001 | Algoritma dan Pemrograman |
| 138572311002 | Algoritma dan Pemrograman |
| NULL     | Pengantar Ilmu Komputer |
| 138572311002 | Pancasila |
| 138572311001 | Pancasila |
| NULL     | Bahasa Indonesia |
| NULL     | Praktikum Basis Data |
| NULL     | Praktikum Algoritma dan Pemrograman |
+-----+-----+
9 rows in set (0.01 sec)
```

Sebagaimana dapat dilihat pada tabel di atas, query tersebut menggabungkan data dari Tabel `ambil_mk` dan Tabel `mata_kuliah` **dengan menampilkan seluruh data pada Tabel `mata_kuliah`**. Pada tabel di atas, **data yang kosong pada Tabel `ambil_mk` menghasilkan keluaran NULL**.

### 7.2.3 Mengenal Operator LEFT OUTER JOIN

Perintah LEFT OUTER JOIN digunakan untuk menggabungkan dua tabel dan **menampilkan semua data pada tabel yang ada di sebelah kiri Operator LEFT OUTER JOIN**. Sebagai contoh, selain Tabel `ambil_mk` dan Tabel `mata_kuliah`, terdapat sebuah tabel yang berisi data dosen yang bernama **Tabel `dosen`**.

```
mysql> SELECT * FROM dosen;
+-----+-----+-----+-----+
| nip    | nama    | alamat | no_hp  |
+-----+-----+-----+-----+
| 198305222013112002 | Ratih   | Malang | 085474423 |
| 198407042016122001 | Fakhriyah | Jember | 0824563672 |
| 198503042002101005 | Agus    | Malang | 085467788 |
| 198608202009091003 | Rudi    | Sidoarjo | 084678769 |
| 198707262011072001 | Putri   | Sidoarjo | 085556777 |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Untuk menampilkan data nama dosen dan kode mata kuliah yang diampunya dan sudah diambil **dengan menampilkan semua data dosen**, kita dapat menggabungkan data Tabel `dosen` dan Tabel `ambil_mk` dengan Operator LEFT OUTER JOIN.

Contoh:

```
SELECT a.nama, b.kode_mk
FROM dosen a
LEFT OUTER JOIN ambil_mk b
ON a.nip = b.nip;
```

Hasil dari query di atas:

```
mysql> SELECT a.nama, b.kode_mk
-> FROM dosen a LEFT OUTER JOIN ambil_mk b
-> ON a.nip=b.nip;
+-----+-----+
| nama    | kode_mk |
+-----+-----+
| Ratih   | 200500123001 |
| Fakhriyah | 200500123004 |
| Fakhriyah | 200500123004 |
| Agus    | NULL      |
| Rudi    | 200500123002 |
| Putri   | 200500123002 |
+-----+-----+
```

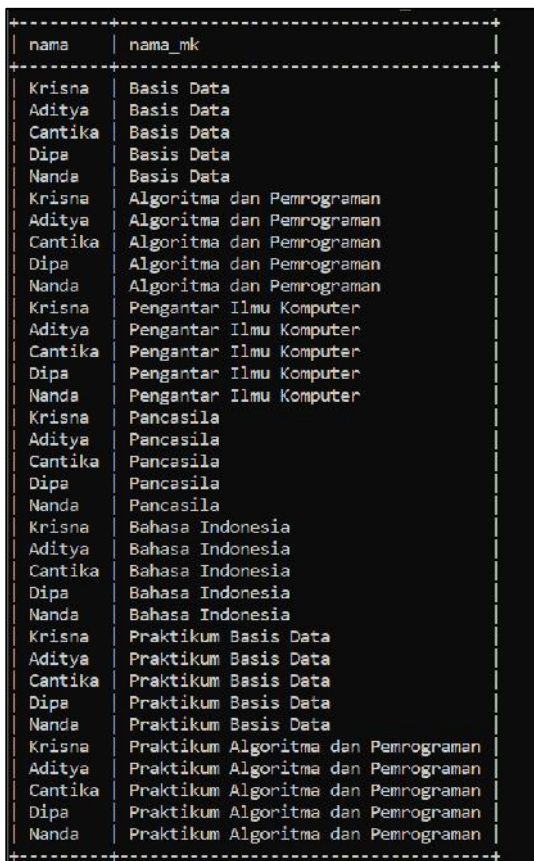
## 7.2.4 Mengenal Perintah CROSS JOIN

Perintah join juga dapat digunakan untuk melakukan perkalian kartesian antara 2 tabel, yaitu dengan menggunakan Operator **CROSS JOIN**.

### Contoh:

Untuk melakukan perkalian kartesian antara **field nama pada Tabel mahasiswa** dan **field nama\_mk pada Tabel mata\_kuliah**, kita dapat menggunakan Operator CROSS JOIN. Dengan demikian, setiap nama mahasiswa akan berpasangan dengan setiap nama mata kuliah yang ada.

```
SELECT a.nama, b.nama_mk  
FROM mahasiswa a  
CROSS JOIN mata_kuliah b;
```



nama	nama_mk
Krisna	Basis Data
Aditya	Basis Data
Cantika	Basis Data
Dipa	Basis Data
Nanda	Basis Data
Krisna	Algoritma dan Pemrograman
Aditya	Algoritma dan Pemrograman
Cantika	Algoritma dan Pemrograman
Dipa	Algoritma dan Pemrograman
Nanda	Algoritma dan Pemrograman
Krisna	Pengantar Ilmu Komputer
Aditya	Pengantar Ilmu Komputer
Cantika	Pengantar Ilmu Komputer
Dipa	Pengantar Ilmu Komputer
Nanda	Pengantar Ilmu Komputer
Krisna	Pancasila
Aditya	Pancasila
Cantika	Pancasila
Dipa	Pancasila
Nanda	Pancasila
Krisna	Bahasa Indonesia
Aditya	Bahasa Indonesia
Cantika	Bahasa Indonesia
Dipa	Bahasa Indonesia
Nanda	Bahasa Indonesia
Krisna	Praktikum Basis Data
Aditya	Praktikum Basis Data
Cantika	Praktikum Basis Data
Dipa	Praktikum Basis Data
Nanda	Praktikum Basis Data
Krisna	Praktikum Algoritma dan Pemrograman
Aditya	Praktikum Algoritma dan Pemrograman
Cantika	Praktikum Algoritma dan Pemrograman
Dipa	Praktikum Algoritma dan Pemrograman
Nanda	Praktikum Algoritma dan Pemrograman

Pada CROSS JOIN, tidak diperlukan adanya foreign key pada kedua tabel sehingga **klausula ON tidak dibutuhkan**.

### 4.3 PRAKTIKUM

1. Pada praktikum ini, kita akan menggunakan basis data bernama “universitas” yang telah kita buat pada Modul 2 Objek basis Data dan Modul 4 Data Management Language (DML). Aktifkan basis data dengan menuliskan sintaks SQL berikut ini:

```
mysql> USE universitas;  
Database changed  
mysql>
```

2. Tampilkan data-data di bawah ini dengan INNER JOIN, RIGHT OUTER JOIN, LEFT OUTER JOIN, dan CROSS JOIN.

- a. Data nama mahasiswa dan kode mata kuliah yang diambilnya.
- b. Data nama mahasiswa dan judul skripsi yang diambilnya.
- c. Data nama mahasiswa dan kode mata kuliah yang diambilnya **dengan menampilkan semua nama mahasiswa.**
- d. Data nim mahasiswa dan nama mata kuliah yang diambilnya **dengan menampilkan semua mata mata kuliah.**
- e. Data nama dosen dan nama mata kuliah yang diampunya dan sudah diambil oleh mahasiswa.
- f. Data semua nama dosen yang dipasangkan dengan semua kode mata kuliah.

**Buat laporan praktikum dalam format .pdf dan ikuti format penamaan file berikut:**

**<nama>\_<NIM>\_<kelas dalam huruf>\_Praktikum 1.pdf**

**Contoh: Ananda\_20040567001\_A\_Praktikum1.pdf**

## MODUL 8

### SUBQUERY

#### 8.1 Bahasan dan Tujuan

##### 8.1.1 Bahasan

Membahas perintah-perintah *subquery* dalam proses pengambilan data dari tabel secara kompleks.

##### 8.1.2 Tujuan

1. Mahasiswa mampu memahami keterhubungan entitas di dalam basis data.
2. Mahasiswa mampu mengimplementasikan perintah operasi subquery dan jenis-jenisnya dalam menyelesaikan kasus-kasus pengambilan data.

#### 8.2 Dasar Teori

##### 8.2.1 SUBQUERY

Subquery atau nested select adalah perintah **SELECT** yang berada di dalam perintah SQL yang lain (**SELECT**, **INSERT**, **UPDATE**, **DELETE**). Perintah *subquery* bermanfaat dalam penyederhanaan *query* pada persoalan yang nilai-nilainya tidak diketahui.

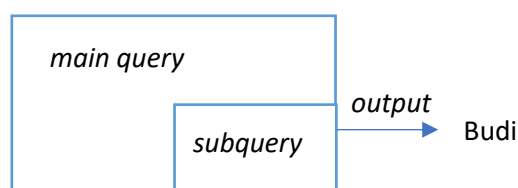
Sintaks formal *subquery* yaitu:

```
SELECT A1, A2, ..., An
FROM r1, r2, r3, ..., rm
WHERE P
      (SELECT A1, A2, ..., An
       FROM r1, r2, r3, ..., rm
       WHERE P)
```

*Subquery* dapat diklasifikasikan ke dalam tiga jenis yaitu:

##### a. *Scalar Subquery*

*Scalar subquery* hanya mengembalikan hasil satu baris data. *Scalar subquery* dapat menggunakan operator baris tunggal =, >, >=, <, <=, atau <>. Apabila dianalogikan *scalar subquery* ditunjukkan oleh gambar di bawah ini.





Masih menggunakan tabel-tabel pada basis data Universitas. Misalkan kita ingin mendapatkan data mahasiswa yang jenis kelaminnya sama dengan nama "Krisna".

```
mysql> SELECT * FROM mahasiswa;
```

nim	nama	jenis_kelamin	alamat
138572311001	Nanda	P	Sidoarjo
138572311002	Dipa	L	Blitar
138624411001	Cantika	P	Blitar
138624411002	Aditya	P	Tulungagung
138752411001	Krisna	P	Surabaya

```
5 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM mahasiswa WHERE jenis_kelamin =  
-> (SELECT jenis_kelamin FROM mahasiswa WHERE nama = "Krisna");
```

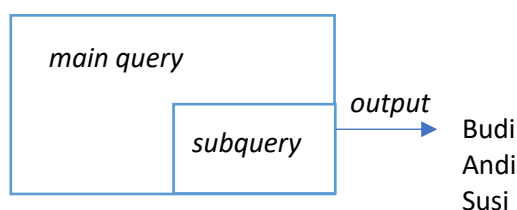
nim	nama	jenis_kelamin	alamat
138572311001	Nanda	P	Sidoarjo
138624411001	Cantika	P	Blitar
138624411002	Aditya	P	Tulungagung
138752411001	Krisna	P	Surabaya

```
4 rows in set (0.01 sec)
```

Berdasarkan *query* dan hasil luaran di atas langkah pertama yang dilakukan adalah mencari jenis kelamin mahasiswa dengan nama "Krisna" kemudian hasilnya yaitu "P" yang digunakan sebagai kata kunci dalam pencarian pada *query* utama.

### b. Multiple-row

*Multiple-row subquery* mengembalikan lebih dari satu baris data. *Multiple-row subquery* ini dapat menggunakan operator komparasi **IN**, **ANY**, **SOME**, atau **ALL**. *Multiple-row subquery* dianalogikan seperti gambar di bawah ini.



### Operator IN

Operator IN memiliki makna **termasuk ke dalam member di dalam list**. Sebagai contoh kita ingin menampilkan data dosen yang mengajar mata kuliah.

```
mysql> SELECT * FROM dosen;
```

NIP	Nama	Alamat	No_HP	tanggal_lahir
198305222013112002	Ratih	Malang	085474423	1983-05-22
198407042016122001	Fakhriyah	Jember	0824563672	1984-07-04
198503042002101005	Agus	Malang	085467788	1985-03-04
198608202009091003	Rudi	Sidoarjo	084678769	1986-08-20
198707262011072001	Putri	Sidoarjo	085556777	1986-07-26

```
5 rows in set (0.01 sec)
```

```
mysql> SELECT * FROM ambil_mk;
```

nim	NIP	kode_mk	nilai
138572311001	198305222013112002	200500123001	85.00
138572311002	198305222013112002	200500123001	70.00
138624411001	198305222013112002	200500123001	80.00
138624411002	198305222013112002	200500123001	83.00
138752411001	198305222013112002	200500123001	80.00
138572311001	198407042016122001	200500123002	70.00
138572311002	198407042016122001	200500123002	85.00

```
7 rows in set (0.01 sec)
```

```
mysql> SELECT NIP, Nama FROM dosen WHERE  
-> NIP IN (SELECT NIP from ambil_mk);
```

NIP	Nama
198305222013112002	Ratih
198407042016122001	Fakhriyah

```
2 rows in set (0.01 sec)
```

Berdasarkan contoh di atas dapat diperhatikan bahwa data dosen yang ditampilkan adalah data dosen yang termasuk di dalam *list* dosen di tabel *ambil\_mk*.

### Operator ANY/SOME

Operator ANY/SOME mempunyai makna membandingkan suatu nilai dengan setiap nilai yang dikembalikan oleh *subquery*.

Operator = ANY ekuivalen dengan IN.  
Operator < ANY ekuivalen dengan MAX (kurang dari maks).  
Operator > ANY ekuivalen dengan MIN (lebih dari min).

Sebagai contoh kita ingin mendapatkan data mata kuliah yang memiliki sks lebih besar dari sembarang sks mata kuliah di semester 1.

```
mysql> SELECT * FROM mata_kuliah;
```

kode_mk	nama_mk	sks	semester
200500123001	Basis Data	3	2
200500123002	Algoritma dan Pemrograman	3	3
200500123003	Pengantar Ilmu Komputer	3	2
200500123004	Pancasila	2	3
200500123005	Bahasa Indonesia	2	1
200500123006	Praktikum Basis Data	1	2
200500123007	Praktikum Algoritma dan Pemrograman	1	1
200500123008	Skripsi	6	7

```
8 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM mata_kuliah  
-> WHERE sks > ANY  
-> (SELECT sks FROM mata_kuliah WHERE semester = 1);
```

kode_mk	nama_mk	sks	semester
200500123001	Basis Data	3	2
200500123002	Algoritma dan Pemrograman	3	3
200500123003	Pengantar Ilmu Komputer	3	2
200500123004	Pancasila	2	3
200500123005	Bahasa Indonesia	2	1
200500123008	Skripsi	6	7

```
6 rows in set (0.00 sec)
```

2, 1

Berdasarkan contoh di atas dapat kita pahami bahwa, data yang akan ditampilkan adalah data mata kuliah yang besaran sks nya **lebih dari nilai minimal** sks yang ada selama semester 1.

### Operator ALL

Operator ALL memiliki arti membandingkan suatu nilai dengan semua nilai yang dikembalikan oleh *subquery*. Misalkan kita ingin menampilkan data mata kuliah yang memiliki sks lebih besar dari semua mata kuliah di semester 2.

```
mysql> SELECT * FROM mata_kuliah  
-> WHERE sks > ALL  
-> (SELECT sks FROM mata_kuliah WHERE semester = 2);
```

kode_mk	nama_mk	sks	semester
200500123008	Skripsi	6	7

```
1 row in set (0.01 sec)
```

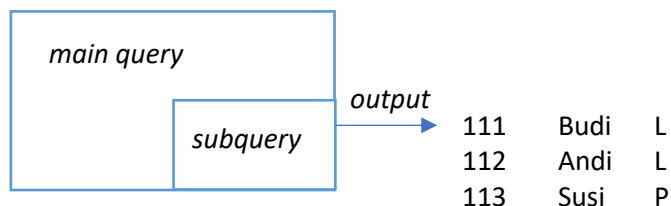
3, 1

Operator < ALL ekuivalen dengan **MIN** (kurang dari min).  
Operator > ALL ekuivalen dengan **MAX** (lebih dari maks).

### c. Multiple-Column

*Multiple-column subquery* mengembalikan lebih dari satu baris dan satu kolom data. Jenis *subquery*

ini dapat diimplementasikan di dalam klausa **WHERE**, **HAVING**, atau **FROM**. Analogi dari *subquery* ini ditunjukkan oleh gambar di bawah ini.



*Multiple-column subquery* juga dapat menggunakan operator komparasi **IN**, **ANY/SOME**, atau **ALL**. Pada *query* ini, nilai atau hasil luaran *subquery* dalam bentuk kolom ganda yang akan dikomparasikan dengan *query* utama.

Sebagai contoh kita ingin menampilkan data mata kuliah yang semester dan sksnya sesuai dengan semester dan sks mata kuliah dengan kode "200500123001".

```
mysql> SELECT * FROM mata_kuliah
-> WHERE (sks,semester) IN
-> (SELECT sks, semester FROM mata_kuliah WHERE kode_mk = "200500123001");
```

kode_mk	nama_mk	sks	semester
200500123001	Basis Data	3	2
200500123003	Pengantar Ilmu Komputer	3	2

2 rows in set (0.01 sec)

sk  
3  
semester  
2

#### d. Operator *EXISTS* dan *NOT EXISTS*

Operator *EXISTS* dan *NOT EXISTS* digunakan untuk memeriksa apakah *subquery* mengembalikan hasil atau tidak. Sebagai contoh, *query* di bawah ini akan mendapatkan data mata kuliah yang diambil oleh mahasiswa.

```
mysql> SELECT * FROM mata_kuliah m
-> WHERE EXISTS (SELECT * FROM ambil_mk a
-> WHERE m.kode_mk = a.kode_mk);
```

kode_mk	nama_mk	sks	semester
200500123001	Basis Data	3	2
200500123002	Algoritma dan Pemrograman	3	3

2 rows in set (0.01 sec)

Sedangkan pernyataan *query* di bawah ini akan menampilkan data mata kuliah yang tidak diambil oleh mahasiswa.

```
mysql> SELECT * FROM mata_kuliah m
-> WHERE NOT EXISTS (SELECT * FROM ambil_mk a
-> WHERE m.kode_mk = a.kode_mk);
```

kode_mk	nama_mk	sks	semester
200500123003	Pengantar Ilmu Komputer	3	2
200500123004	Pancasila	2	3
200500123005	Bahasa Indonesia	2	1
200500123006	Praktikum Basis Data	1	2
200500123007	Praktikum Algoritma dan Pemrograman	1	1
200500123008	Skripsi	6	7

```
6 rows in set (0.00 sec)
```

#### e. *SUBQUERY* dan Fungsi AGREGAT

Perintah-perintah *subquery* juga dapat melibatkan fungsi-fungsi agregat. Sebagai contoh kita ingin mendapatkan data mata kuliah yang memiliki sks sama dengan sks terkecil.

```
mysql> SELECT * FROM mata_kuliah WHERE sks =
-> (SELECT MIN(sks) FROM mata_kuliah);
```

kode_mk	nama_mk	sks	semester
200500123006	Praktikum Basis Data	1	2
200500123007	Praktikum Algoritma dan Pemrograman	1	1

```
2 rows in set (0.03 sec)
```

#### f. *SUBQUEY* dan *JOIN*

Pada beberapa kasus sederhana, fungsionalitas *subquery* dengan *join* dapat dipertukarkan. Dimana keduanya dapat digunakan untuk menyelesaikan persoalan yang sama. Sebagai contoh, misalkan kita ingin mendapatkan kode dosen dan nama dosen yang tidak mengajar mata kuliah.

Pendekatan *subquery*:

```
mysql> SELECT NIP, nama FROM dosen WHERE NIP NOT IN
-> (SELECT NIP FROM ambil_mk);
```

NIP	nama
198503042002101005	Agus
198608202009091003	Rudi
198707262011072001	Putri

```
3 rows in set (0.03 sec)
```

Pendekatan *join*:

```
mysql> SELECT d.NIP, nama FROM dosen d LEFT OUTER JOIN ambil_mk a
-> ON d.NIP = a.NIP WHERE a.NIP IS NULL;
```

NIP	nama
198503042002101005	Agus
198608202009091003	Rudi
198707262011072001	Putri

```
3 rows in set (0.01 sec)
```

### **8.3 TUGAS PRAKTIKUM**

1. Tampilkan data mahasiswa yang alamatnya sama dengan mahasiswa dengan nama "Dipa". Data yang ditampilkan tidak termasuk data mahasiswa tersebut.
2. Tampilkan nim, nama, dan alamat mahasiswa yang tempat tinggalnya sama dengan dosen yang mengajar matakuliah dengan semester lebih kecil dari sembarang semester.

## MODUL 9

### SQL View & Trigger

#### 6.1 Bahasan dan Tujuan

##### 6.1.1 Bahasan

Membahas implementasi View sebagai tabel virtual dan Trigger sebagai kumpulan kode SQL yang berjalan secara otomatis untuk mengeksekusi perintah INSERT, UPDATE, DELETE.

##### 6.1.2 Tujuan

- Memahami konsep dasar view di dalam basis data
- Memahami implementasi view, termasuk algoritma dan jenis-jenisnya yang tersedia
- Mampu menyelesaikan kasus-kasus pengambilan data dengan menggunakan pendekatan view
- Memahami konsep dasar trigger di dalam basis data.
- Memahami implementasi trigger sebagai bentuk respon atas suatu kejadian.
- Mampu menyelesaikan kasus-kasus manipulasi data yang kompleks dengan memanfaatkan trigger.

#### 6.2 Dasar Teori VIEW

VIEW adalah perintah untuk membuat table virtual yang menyimpan kode SQL. Dengan view kita bisa membuat kode SQL yang kompleks dikemas menjadi satu table sederhana. View akan menyimpan kode SQL yang kompleks tadi menjadi single table virtual yang lebih mudah untuk digunakan. Pada modul ini gunakan database ViewdanTrigger.

Sintaks VIEW sebagai berikut:

```
CREATE VIEW <nama view> AS Kode SQL
```

Contoh:

```
CREATE VIEW pengampu_matkul AS Kode SQL
```

Saat kita mengeksekusi CREATE VIEW maka akan terbentuk table virtual yang menyimpan kode SQL.

Contoh, membuat kode SQL yang menghubungkan tabel dosen dan tabel mata\_kuliah secara INNER JOIN dan menyimpannya ke view.

```
mysql> SELECT Mata_Kuliah.nama_dosen, Nama_mk, sks, total_jam_kerja FROM Mata_Kuliah INNER JOIN dosen ON
Mata_Kuliah.nama_dosen= dosen.nama_dosen; #NIM
+-----+-----+-----+-----+
| nama_dosen | Nama_mk          | sks | total_jam_kerja |
+-----+-----+-----+-----+
| Anggi      | Pancasila        | 1   | 7                |
| Benny     | Bahasa Arab I    | 1   | 6                |
| Cherill    | Calculus         | 3   | 7                |
| Davi      | Kewarganegaraan | 2   | 5                |
| Ega       | Teosofi          | 3   | 6                |
| Fachri    | Digital Electronic | 3   | 5                |
| Gaga      | Bahasa Inggris I | 2   | 2                |
+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

Dengan view kita bisa membuat table virtual yang menyimpan query join di atas

```
mysql> CREATE VIEW pengampu_matkul AS
-> SELECT Mata_Kuliah.nama_dosen, Nama_mk, sks, total_jam_kerja FROM Mata_Kuliah INNER JOIN dosen ON
Mata_Kuliah.nama_dosen= dosen.nama_dosen;#NIM
Query OK, 0 rows affected (0.26 sec)

mysql> Select * FROM pengampu_matkul ;#NIM
+-----+-----+-----+-----+
| nama_dosen | Nama_mk          | sks | total_jam_kerja |
+-----+-----+-----+-----+
| Anggi      | Pancasila        | 1   | 7                |
| Benny     | Bahasa Arab I    | 1   | 6                |
| Cherill    | Calculus         | 3   | 7                |
| Davi      | Kewarganegaraan | 2   | 5                |
| Ega       | Teosofi          | 3   | 6                |
| Fachri    | Digital Electronic | 3   | 5                |
| Gaga      | Bahasa Inggris I | 2   | 2                |
+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

Fungsi lain setelah membuat view, misalkan ingin query nama dosen yang mengampu mata kuliah Teosofi

```
mysql> SELECT nama_dosen, nama_mk FROM pengampu_matkul WHERE nama_mk="teosofi";#NIM
+-----+-----+
| nama_dosen | Nama_mk |
+-----+-----+
| Ega       | Teosofi |
+-----+-----+
1 row in set (0.00 sec)
```

Untuk menghapus VIEW menggunakan sintaks sebagai berikut:

DROP VIEW <nama\_view>;

```
mysql> DROP VIEW pengampu_matkul;#NIM
Query OK, 0 rows affected (0.13 sec)
```

### 6.3 Dasar teori Trigger

TRIGGER adalah kumpulan kode SQL yang berjalan secara otomatis untuk mengeksekusi perintah INSERT, UPDATE, DELETE.



Biasanya TRIGGER akan dijalankan sebelum atau sesudah proses INSERT, UPDATE, DELETE (Perintah DML)

Sintaks TRIGGER sebagai berikut:

```
DELIMITER $$
```

```
CREATE TRIGGER nama_trigger
```

```
{BEFORE | AFTER} {INSERT | UPDATE| DELETE }
```

```
ON nama_table
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    KODE SQL
```

```
END$$
```

```
DELIMITER ;
```

Untuk memulai menggunakan TRIGGER kita gunakan CREATE TRIGGER dilanjutkan nama TRIGGER yang ingin dibuat {BEFORE | AFTER} adalah waktu TRIGGER akan dijalankan, apakah sebelum atau sesudah database dimodifikasi oleh perintah DML {INSERT | UPDATE | DELETE} adalah perintah DML yang mengaktifkan TRIGGER, penggunaan detail waktu TRIGGER akan dijelaskan pada tabel berikut:

No	Waktu TRIGGER	Keterangan TRIGGER
1	BEFORE INSERT	TRIGGER dijalankan sebelum record dimasukkan ke database
2	AFTER INSERT	TRIGGER dijalankan sesudah record dimasukkan ke database
3	BEFORE UPDATE	TRIGGER dijalankan sebelum record dirubah di database
4	AFTER UPDATE	TRIGGER dijalankan

		sesudah record dirubah database
5	BEFORE DELETE	TRIGGER dijalankan sebelum record dihapus di database
6	AFTER DELETE	TRIGGER dijalankan sesudah record dihapus di database

ON mendefinisikan table yang mengaktifkan TRIGGER

BEGIN END adalah pernyataan yang membungkus kode TRIGGER

Pastikan diawal gunakan DELIMITER \$\$ dan diakhir dikembalikan ke DELIMITER.

Contoh kasus menggunakan database ViewdanTrigger dan tabel mata\_kuliah.

Table mata\_kuliah -> menyimpan data mata kuliah

Table log\_ mata\_kuliah -> menyimpan perubahan data mata kuliah

Jadi setiap ada perubahan data (UPDATE) nama mata kuliah pada tabel mata\_kuliah maka akan disimpan di table log\_ mata\_kuliah tentang histori perubahan data tersebut. Dengan adanya log perubahan data mata\_kuliah maka akan memudahkan dalam melihat histori data mata\_kuliah yang pernah berubah dalam sistem.

Membuat tabel log\_ mata\_kuliah

```
mysql> create table log_Mata_Kuliah (id_log INT (10) AUTO_INCREMENT, Kode_mk varchar (12), Nama_mk_lama char(25), Nama_mk_baru char(25), waktu DATE, PRIMARY KEY(id_log));#NIM
Query OK, 0 rows affected, 1 warning (0.81 sec)
```

Membuat TRIGGER, sintak trigger ini menyimpan data perubahan nama mata kuliah sebelum perintah UPDATE dijalankan

```
mysql> DELIMITER $$
mysql> CREATE TRIGGER update_nama_matkul
-> BEFORE UPDATE
-> ON mata_kuliah
-> FOR EACH ROW
-> BEGIN
-> INSERT INTO log_Mata_Kuliah
-> set Kode_mk = OLD.Kode_mk,
-> Nama_mk_lama=old>Nama_mk,
-> Nama_mk_baru=new>Nama_mk,
-> waktu = NOW();
-> END$$
Query OK, 0 rows affected (0.65 sec)
```

Tips: untuk menampilkan mysql> gunakan “DELIMITER ;” pada akhir perintah.

Keyword OLD digunakan untuk mengambil data kolom di table yang lama sedangkan keyword NEW digunakan untuk mengambil data kolom di table yang baru.

Tabel sebelum diupdate:

```
mysql> SELECT * FROM mata_kuliah;#NIM
+-----+-----+-----+-----+-----+
| Kode_mk | Nama_mk          | sks | semester | nama_dosen |
+-----+-----+-----+-----+-----+
| MKP1011 | Calculus         | 3   | 3         | Cherill    |
| MKP1213 | Kewarganegaraan | 2   | 5         | Davi       |
| MKP6789 | Bahasa Arab I   | 1   | 3         | Benny     |
| MKU1415 | Teosofi         | 3   | 5         | Ega       |
| MKU2345 | Pancasila       | 1   | 3         | Anggi     |
| MKW1617 | Digital Electronic | 3   | 5         | Fachri    |
| MKW1819 | bahasa inggris 1 | 2   | 5         | Gaga      |
+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

Sekarang kita akan coba update nama matakuliah dengan kode mk MKW1819. Sebelum diupdate nama mata kuliah dengan kode mk MKW1819 adalah “bahasa inggris 1” sekarang akan diganti menjadi “bahasa indonesia”

```
mysql> UPDATE mata_kuliah
-> SET Nama_mk = 'bahasa indonesia'
-> WHERE Kode_mk = 'MKW1819';
->
-> DELIMITER ;#NIM
Query OK, 1 row affected (0.99 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

Sekarang coba lakukan perintah SELECT untuk melihat table log\_mata\_kuliah

```
mysql> SELECT * FROM log_mata_kuliah;#NIM
+-----+-----+-----+-----+-----+
| id_log | Kode_mk | Nama_mk_lama      | Nama_mk_baru      | waktu      |
+-----+-----+-----+-----+-----+
|      1 | MKW1819 | bahasa inggris 1 | bahasa indonesia  | 2022-11-06 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Untuk menghapus trigger

```
mysql> DROP TRIGGER update_nama_matkul;
-> DELIMITER ;#NIM
Query OK, 0 rows affected (0.23 sec)
```

6.2.1.

Tugas:

1. Buatlah laporan praktikum untuk mendokumentasikan langkah-langkah praktikum Mulai dari pembuatan “SQL View & Trigger” pada modul yang telah dilakukan, Sertakan gambar dari masing-masing langkah, beserta penjelasan singkat.
2. Buatlah tabel mahasiswa pada database viewdantrigger:

Nim	Nama_mahasiswa	semester	nama_dosen
12345	Hanzel	3	Anggi
12346	August	3	Benny
12347	Sea	3	Cherill
12348	Adam	5	Davi
12349	Elea	5	Ega
123410	Rafa	5	Fachri
123411	Kala	5	Gaga

3. Membuat view dari relasi antara tabel “dosen”, “mahasiswa” dan “matakuliah” untuk menampilkan data mengajar dosen dari database viewdantrigger dengan nama "view\_mengajar". Tampilkan NIP, nama dosen, nama mahasiswa, nama mk dan semester

4. Membuat view dari relasi antara tabel “dosen”, “mahasiswa” dan “matakuliah” untuk menampilkan data mengajar dosen dari database view dan trigger dengan nama "view\_mengajar". Tampilkan NIP, nama dosen, nama mahasiswa, nama mk dan semester.
5. Tampilkan view yang ada di database aktif!
6. Buatlah tabel log mahasiswa! Gunakan nim sebagai primary key, cukup membuat nim, nama mahasiswa dan waktu.
7. Membuat trigger dengan nama "update nama mahasiswa".
8. Ubah nim 12347 dengan nama “Sea Utama”
9. Ubah nim 12311 dengan nama “Aska Kala”
10. Tampilkan hasil log Mahasiswa!

Kumpulkan laporan dalam format PDF dengan penamaan file:

<nama>\_<NIM>\_<kelas dalam huruf>\_Praktikum 10.pdf

Contoh: Ananda\_20040567001\_A\_Praktikum6.pdf

## MODUL 10

### FUNGSI DAN STORED PROCEDURE

#### 10.1 Bahasan dan Tujuan

##### 10.1.1 Bahasan

Membahas tentang implementasi fungsi dan stored procedure pada MySQL

##### 10.1.2 Tujuan

1. Mahasiswa mampu memahami dan mengimplementasikan fungsi dan stored procedure pada MySQL.
2. Mahasiswa mampu memahami dan mengimplementasikan ekspresi pada MySQL.
3. Mahasiswa mampu memahami dan mengimplementasikan looping dengan MySQL.

#### 10.2 Dasar Teori

##### 10.2.1 Fungsi pada MySQL

Fungsi SQL adalah sekumpulan query biasanya query yang detail dan panjang yang dibungkus menjadi satu dan disimpan dalam database dan kemudian apabila diperlukan hanya tinggal mengaksesnya tanpa menyetikkan query detail.

Secara umum, ada beberapa faktor yang perlu diperhatikan pada saat membuat fungsi SQL, antara lain:

1. Nama fungsi
2. Nomor dan nama argument
3. Tipe data dari setiap argument
4. Tipe dari hasil fungsi
5. Fungsi action

Struktur umum dari fungsi SQL adalah sebagai berikut:

```
CREATE FUNCTION nama_fungsi([arg] tipe_data (ukuran_data), [arg1] tipe_data1 (ukuran_data1), ...)
```

```
RETURNS tipe_data_hasil (ukuran_data)
```

```
DETERMINISTIC
```

```
RETURN badan fungsi (diakhiri tanda “;”)
```

Berikut ini adalah contoh sederhana dari fungsi SQL dengan 1 argumen:

```
mysql> CREATE FUNCTION hello(a varchar(10))
  -> RETURNS varchar(20)
  -> DETERMINISTIC
  -> RETURN concat('Hello, ',a,'!');
Query OK, 0 rows affected (0.01 sec)
```

Untuk memanggil fungsi yang sudah dibuat, dapat digunakan perintah SELECT. Contohnya:

```
SELECT hello('Narendra');
```

```
mysql> SELECT hello('Narendra');
+-----+
| hello('Narendra') |
+-----+
| Hello, Narendra! |
+-----+
1 row in set (0.00 sec)
```

Contoh fungsi SQL yang menggunakan 3 argumen adalah:

```
mysql> CREATE FUNCTION func_perkalian(angka1 float, angka2 float, angka3 float)
  -> RETURNS float
  -> DETERMINISTIC
  -> RETURN angka1*(angka2+angka3);
Query OK, 0 rows affected (0.01 sec)
```

Untuk memanggil fungsi func\_perkalian, dapat digunakan perintah SELECT, contohnya:

```
SELECT func_perkalian(5, 2, 10) AS Hasil;
```

```
mysql> SELECT func_perkalian(5, 2, 10) AS HASIL;
+-----+
| HASIL |
+-----+
|     60 |
+-----+
1 row in set (0.00 sec)
```

Untuk menghapus fungsi, kita dapat menggunakan keyword **DROP FUNCTION**. Contohnya:

```
mysql> DROP FUNCTION hello;
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT hello('Anindya');
ERROR 1305 (42000): FUNCTION universitas.hello does not exist
mysql>
```

## 10.2.2 Stored Procedure pada MySQL

Berikut struktur pembuatan stored procedure pada MySQL.

### 1. Pembuatan stored procedure:

Delimiter //

```
CREATE PROCEDURE nama_stored_procedure ([arg1 tipe_data1(ukuran_data1) , ... ])
```

```
BEGIN
```

Badan stored procedure (diakhiri dengan tanda ';' )

```
END //
```

Di bawah ini adalah contoh stored procedure dengan 1 argumen:

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE nama_mahasiswa(jk1 char(1))
-> BEGIN
-> SELECT nama FROM mahasiswa WHERE jk=jk1;
-> END //
Query OK, 0 rows affected (0.01 sec)
```

Untuk memanggil stored procedure, dapat digunakan keyword CALL. Contohnya:

```
mysql> CALL nama_mahasiswa('L');
+-----+
| nama |
+-----+
| Dipa |
| Aditya |
| Krisna |
+-----+
3 rows in set (0.01 sec)
Query OK, 0 rows affected (0.02 sec)
```

### 2. Menghapus Stored Procedure

Untuk menghapus stored procedure, dapat digunakan perintah berikut ini:

```
DROP PROCEDURE nama_stored_procedure;
```



Contoh :

```
mysql> DROP PROCEDURE nama_mahasiswa;
Query OK, 0 rows affected (0.01 sec)

mysql> CALL nama_mahasiswa('L');
ERROR 1305 (42000): PROCEDURE universitas.nama_mahasiswa does not exist
```

### 6.2.3 Ekspresi pada Fungsi MySQL

MySQL juga mengenal ekspresi untuk melakukan seleksi kondisi atau yang biasa dikenal dengan perintah IF THEN ELSE. Berikut ini adalah contoh penggunaan ekspresi untuk menghitung harga total fotokopi berdasarkan jumlah lembarannya:

```
mysql> DELIMITER //
mysql> CREATE FUNCTION fotokopi(lembar int)
-> RETURNS int
-> DETERMINISTIC
-> BEGIN
-> DECLARE harga_total int;
-> IF lembar<=100 THEN SET harga_total=lembar*250;
-> ELSEIF LEMBAR <=200 THEN SET harga_total=lembar*200;
-> ELSE SET harga_total=lembar*175;
-> END IF;
-> RETURN harga_total;
-> END //
Query OK, 0 rows affected (0.01 sec)
```

### 6.2.4 Looping pada fungsi MySQL

MySQL juga mengenal perintah untuk melakukan perulangan, yaitu LOOP. Berikut ini adalah contoh fungsi dengan perulangan dalam bahasa MySQL untuk menghitung hasil dari operasi faktorial:

```
mysql> DELIMITER //
mysql> CREATE FUNCTION faktorial(angka int)
-> RETURNS int
-> DETERMINISTIC
-> BEGIN
-> DECLARE hasil int DEFAULT 1;
-> DECLARE hitung int DEFAULT 1;
-> loop_label: LOOP
-> IF hitung>angka THEN LEAVE loop_label;
-> END IF;
-> SET hasil = hasil*hitung;
-> SET hitung = hitung+1;
-> END LOOP;
-> RETURN hasil;
-> END //
Query OK, 0 rows affected (0.01 sec)
```

### 6.3 Tugas

1. Buatlah sebuah stored procedure MySQL untuk menampilkan data nama mata kuliah dengan kode mata kuliah tertentu!
2. Buatlah sebuah fungsi MySQL untuk menghitung luas segitiga!
3. Buatlah sebuah fungsi MySQL untuk menghitung tarif listrik dengan ketentuan sebagai berikut:

Penggunaan listrik <100 kWh, tarifnya Rp2000,00 / kWh

Penggunaan listrik >100 kWh, tarifnya Rp1000,00/kWh

Dokumentasikan langkah-langkah praktikum dan hasil tugas yang Anda buat dalam sebuah laporan. Kumpulkan laporan dalam format PDF dengan penamaan file:

<nama>\_<NIM>\_<kelas dalam huruf>\_Praktikum10.pdf

Contoh: Ananda\_20040567001\_A\_Praktikum10.pdf