**LAPORAN RESMI**

**MODUL IV**

**SQL**

**SISTEM MANAJEMEN BASIS DATA**



**NAMA :**

**N.R.P :**

**DOSEN :**

**ASISTEN :**

**TGL PRAKTIKUM :**

**Disetujui : … ..……… 2024**

**Asisten**

**LABORATORIUM BISNIS INTELIJEN SISTEM**

**PRODI SISTEM INFORMASI**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

# **BAB I PENDAHULUAN**

## **Latar Belakang**

MySQL adalah sistem manajemen basis data relasional yang populer dan open source. MySQL menggunakan bahasa SQL (Structured Query Language) untuk mengelola data dalam tabel dua dimensi dengan baris dan kolom. MySQL dapat dioperasikan dalam mode client-server, di mana server MySQL berjalan di server dan berbagai program serta library berjalan di sisi klien untuk mengakses dan memanipulasi data. MySQL mendukung beberapa pengguna untuk mengakses database secara bersamaan. Ini memungkinkan banyak pengguna untuk mengakses dan memanipulasi data secara bersamaan. MySQL juga menawarkan fitur keamanan untuk mengamankan akses ke database, termasuk otentikasi pengguna dan enkripsi data.

Pengelolaan database menggunakan DBMS telah memengaruhi banyak aspek kehidupan masyarakat modern, mulai dari bisnis hingga layanan kesehatan dan pemerintahan. Perangkat lunak DBMS telah memudahkan penyimpanan, pencarian, dan manipulasi data dalam skala besar, dan telah membuka peluang baru untuk analisis data dan pengambilan keputusan yang lebih baik. MySQL telah menjadi salah satu pilihan teratas untuk pengelolaan basis data dalam skala kecil hingga besar.

Bahasa SQL (Structured Query Language) adalah bahasa standar untuk mengelola basis data relasional. SQL digunakan untuk membuat, mengelola, dan mengambil data dari basis data dan menjadi bahasa yang sangat penting dalam pengembangan aplikasi dan analisis data. Dalam dunia bisnis, SQL digunakan untuk memproses dan menganalisis data untuk mengambil keputusan bisnis yang lebih baik. Selain itu, SQL juga digunakan di berbagai industri, seperti keuangan, pemerintahan, perbankan, dan banyak lagi.

Praktikum bahasa SQL sangat penting bagi mahasiswa atau siapa saja yang ingin mempelajari bagaimana menggunakan SQL untuk mengelola basis data. Dalam praktikum ini, peserta akan mempelajari konsep dasar SQL, seperti membuat tabel, mengakses data, memanipulasi data, dan membuat kueri. Dengan mempelajari bahasa SQL, peserta dapat meningkatkan keterampilan dalam pengolahan data dan mengambil keputusan bisnis yang lebih baik.

Praktikum bahasa SQL juga sangat penting bagi para profesional TI, analis data, dan administrator basis data. Mereka harus memiliki keterampilan dalam pengelolaan basis data dan SQL agar dapat mengoptimalkan pengambilan data dan pemrosesan data dalam database. Selain itu, mereka harus memahami bagaimana membangun relasi antar tabel untuk mengoptimalkan kinerja database.

Dalam praktikum bahasa SQL, peserta akan belajar cara menggunakan SQL untuk mengakses, memanipulasi, dan mengelola basis data. Mereka akan mempelajari konsep dasar SQL, membuat tabel, mengakses data, memanipulasi data, dan membuat kueri kompleks. Dengan menguasai bahasa SQL, peserta akan memiliki keunggulan kompetitif dalam dunia kerja dan dapat meningkatkan efisiensi dan efektivitas pengambilan keputusan bisnis.

## **Tujuan**

1. Mampu memahami dan membuat perintah SQL lebih dalam dengan berbagai kondisi, Ekpresi, pengurutan dan mampu mengabungkan beberapa argument / perintah pada beberapa tabel (join) dalam basis data.

# **BAB II DASAR TEORI**

## **Dasar Teori**

### SQL

Secara umum perintah-perintah yang terdapat di dalam SQL, diklasifikasikan menjadi tiga bagian, antara lain yaitu :

1. DDL (Data Definition Language)

* Merupakan perintah SQL yang berkaitan dengan pendefinisian suatu struktur database, dalam hal ini database dan table.
* Perintah DDL adalah: CREATE, ALTER, RENAME, DROP.

1. DML (Data Manipulation Language)

* Merupakan perintah SQL yang berkaitan dengan manipulasi atau pengolahan data atau record dalam table.
* Perintah DML antara lain: SELECT, INSERT, UPDATE, DELETE.

1. DCL (Data Control Language)

* Merupakan perintah SQL yang berkaitan dengan manipulasi user dan hak akses (priviledges).
* Perintah SQL yang termasuk dalam DCL antara lain: GRANT, REVOKE.

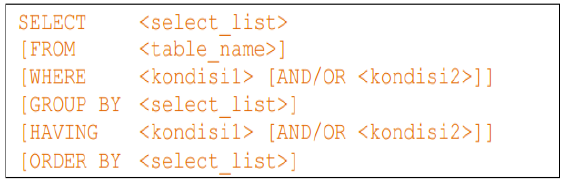
### SELECT

SELECT merupakan salah satu pondasi dalam SQL Programming. SELECT digunakan untuk menampilkan data, terlebih untuk mencari informasi dalam kumpulan data.

* **Sintak**

SELECT dibagi kedalam 6komponen, antara lain :

1. SELECT. Diikuti oleh **<select\_list>,** dapat berupa literal\_value atau column\_list atau asterisk (\*).
2. FROM. Diikuti oleh **<table\_name>** sesuai dengan column\_list. Jadi jika ada data yang diambil dari kolom tertentu, harus diketahui kolom tersebut diambil dari tabel mana. Tabel pada FROM dapat diikuti dengan alias untuk mempermudah penulisan khususnya ketika join dan subquery.
3. WHERE. Diikuti oleh kondisi secara umum.
4. GROUP BY. Diikuti oleh <select\_list>. Bagian ini muncul ketika ada fungsi-fungsi agregasi.
5. HAVING. Diikuti oleh kondisi hanya untuk fungsi -fungsi agregasi.
6. ORDER BY. Diikuti oleh <select\_list>

* **Perintahnya :**

1. **Menyaring Data**

Tidak semua data yang ada pada tabel, ingin ditampilkan. Terlebih ketika tabel terbagi kedalam banyak kolom dengan jumlah data yang sangat besar.

* **Operator Pembanding :**



* **Perintahnya :**

**Select \* From Nama\_Table Where Nama\_Field [Operator Relasional] Ketentuan;**

Contoh :

**Select \* from customer where cust\_id = ‘10003’ or cust\_name =’wascals;**

### Pengurutan Data (ACS, DESC, ORDER BY)

1. Mengurutkan Data

Untuk mengurutkan tampilan data dari suatu table, digunakan klausa Order By. Klausa Order By, dapat digunakan untuk mengurutkan data :

* Asc (Ascending ) : Untuk mengurutkan data dari kecil ke besar
* Desc (Descending) : Untuk mengurutkan data dari besar ke kecil

Perintahnya :

**Select \* From Nama\_Table Order By Nama\_Field\_Key Asc/Desc;**

Contoh :

**Select \* From products Order By prod\_name Asc;**

1. **Operator Between, In, Like**
2. Operator Between

Operator Between merupakan operator yang digunakan untuk menangani operasi jangkauan. Perintahnya :

**Select \* From Nama\_Table Where Nama\_Field\_ketentuan Between 'Ketentuan\_1' And 'Ketentuan\_2';**

Contoh :

**Select \* From orderitems Where quantity Between '1' And '10';**

1. Operator In

Operator In merupakan operator yang digunakan untuk mencocokan suatu nilai. Perintahnya :

**Select Nama\_Field From Nama\_Table Where Nama\_Field\_Pencocok In ('Isi\_Field\_1','Isi\_Field\_2');**

Contoh :

**Menampilkan nama customer, alamat dan email customer tertentu.**

**Select cust\_name,cust\_address,cust\_email From customers Where cust\_id In ('10002','10005');**

1. Operator Like

Operator Like merupakan operator yang digunakan untuk mencari suatu data (search). Perintahnya :

**Select \* From Nama\_Table Where Nama\_Field\_Dicari Like '%Key';**

Contoh :

Select \* From Products Where prod\_name Like '%s';

**Query yang pertama menampilkan produk dengan nama produk diawali huruf dan pada query yang kedua nama produk diakhiri huruf s.**

### Ekspresi Query

Ekspresi Query dapat digunakan untuk melakukan perubahan terhadap field kolom keluaran, menambah baris teks field keluaran.

1. Mengganti Nama Field keluaran, Perintahnya :

**Select Nama\_Field\_Asal As 'Nama\_Field\_Pengganti' From Nama\_Table;**

Contoh :

**Select Kode\_Mtkul As 'Kode Matakuliah', Nama\_Mtkul As 'Matakuliah' From Mtkul;**

1. Menambahkan Baris Teks Field Keluaran, Perintahnya :

**Select 'Nama Field Tambahan', Nama\_Field\_Asal From Nama\_Table;**

Contoh :

**Select vend\_name,’diproduksi di’, vend\_city From vendors;**

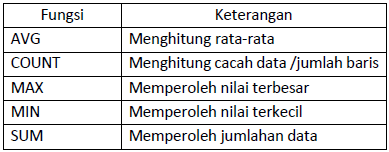
1. Ekspresi Kondisi, Perintahnya :

**Select Nama\_Field\_1 Case Nama\_Field\_2 When 'Nilai\_field\_2' Then 'Keterangan\_1' Else 'Keterangan\_2' End As Nilai\_field\_2 From Nama\_Table;**

**Contoh :**

**Select Kode\_Mtkul, Nama\_Mtkul, Case Sks When '1' Then 'Praktikum' Else 'Matakuliah' End As Sks From Mtkul;**

### Agregasi Dan Pengelompokan Data

Dalam pemrosesan data mentah menjadi data statistik, diperlukan fungsi-fungsi yang dapat meng-agregasi data-data tersebut. Fungsi-fungsi ini meliputi SUM, MIN, MAX, COUNT, dan AVG.

### Multiple-Table Query

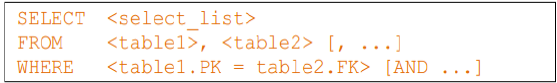
Data-data yang tersimpan dalam basis data, tersebar kedalam beberapa tabel. Tabel-tabel ini dihubungkan dengan yang namanya referential constraint, yaitu hubungan antara foreign key dan primary key.

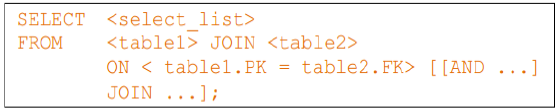
Karena itulah, untuk mendapatkan informasi yang tersebar, dibutuhkan metode untuk menggabungkan property tabel-tabel tersebut. Metode yang digunakan ada 2 macam, yaitu join dan subquery.

Perbedaannya sederhana, join menggunakan satu SELECT, sedangkan subquery menggunakan dua atau lebih SELECT (umumnya dikatakan sebagai SELECT within a SELECT).

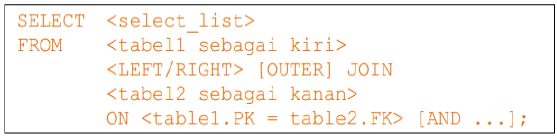
* **Join**

Bentuk join pertama kali adalah menggunakan kata kunci WHERE untuk melakukan penggabungan tabel.



Perkembangan SQL ANSI

* **Tipe Join**

Ada 2 tipe join, yaitu inner join yang lebih menekankan pada keberadaan data yang sama, dan outer join.

Pada INNER JOIN atau CROSS JOIN output/hasil yang ditampilkan adalah data-data dari semua table yang terlibat dimana baris yang tampil hanya yang memiliki kondisi kesamaan data. Kesamaan data berdasarkan relasinya (kesamaan data foreign key dengan primary key tabel yang diacu).

Berikut adalah bentuk umum INNER JOIN yang umumnya hanya disebut sebagai JOIN :

**SELECT nm\_tabel1.nm\_kolom1, nm\_tabel1.nm\_kolom2, nm\_tabel2.nm\_kolom1, nm\_tabel2.nm\_kolom2 FROM tabel1, tabel2 WHERE tabel1.nama\_kolom1**

**(primary key)=tabel2.nama\_kolom1(foreign key yg mengacu ke tabel1)**

Contoh penggunaan Join, kita lihat kembali skema order entry dibawah ini. Menampilkan prod\_name, vend\_name dari table vendors dan products.

Select vendors.vend\_name,products.prod\_name from vendors, products Where vendors.vend\_id = products.vend-id

1. Clausa Join On Alias

**SELECT a.nm\_kolom1, b.nm\_kolom2, a.nm\_kolom3**

**FROM tabel1 a**

**JOIN tabel2 b**

**ON a.nama\_kolom1(primary key)=b.nama\_kolom1(foreign key yg mengacu ke tabel1)**

**WHERE kondisi;**

1. JOIN 3 TABLE ATAU LEBIH

Pada prinsipnya sama, hanya jumlah tabel ditambah dan sintaks disesuaikan. Contoh penerapan join dua tabel atau lebih untuk menampilkan nama customer, tgl order dan total jumlah order.

**select a.cust\_name,b.order\_date,c.quantity from customers a join orders b on a.cust\_id=b.cust\_id join orderitems c on b.order\_num=c.order\_num;**

1. OUTER JOIN

Pada OUTER JOIN hasil yang ditampilkan adalah data-data dari semua tabel yang terlibat baik yang hanya yang memiliki kondisi kesamaan data berdasarkan relasinya (kesamaan data foreign key dengan primary key tabel yang diacu) maupun data yang tidak memiliki kesamaan data berdasarkan relasinya dari salah satu tabel. Terdapat dua tipe OUTER JOIN, yaitu : LEFT OUTER JOIN atau biasa disebut left join dan RIGHT OUTER JOIN atau biasa disebut righ join.

1. LEFT JOIN

Pada LEFT JOIN output/hasil yang ditampilkan adalah data-data dari semua tabel yang terlibat baik yang hanya yang memiliki kondisi kesamaan data berdasarkan relasinya (kesamaan data foreign key dengan primary key tabel yang diacu) maupun data-data yang tidak memiliki kesamaan data berdasarkan relasinya dari tabel sebelah kiri dari klausa LEFT JOIN. Berikut adalah bentuk umum :

**SELECT nm\_tabel1.nm\_kolom1, nm\_tabel1.nm\_kolom2, nm\_tabel2.nm\_kolom1, nm\_tabel2.nm\_kolom2**

**FROM tabel1**

**LEFT JOIN tabel2**

**ON tabel1.nama\_kolom1(primary key)=tabel2.nama\_kolom1(foreign key yg mengacu ke tabel1)**

**WHERE kondisi;**

Contoh :

**select a.cust\_name,b.order\_date**

**from customers a left join orders b on a.cust\_id=b.cust\_id RIGHT JOIN**

Pada RIGHT JOIN output/hasil yang ditampilkan adalah data-data dari semua tabel yang terlibat baik yang hanya yang memiliki kondisi kesamaan data berdasarkan relasinya (kesamaan data foreign key dengan primary key tabel yang diacu) maupun data-data yang tidak memiliki kesamaan data berdasarkan relasinya dari tabel sebelah kanan dari klausa RIGHT JOIN.

**SELECT nm\_tabel1.nm\_kolom1, nm\_tabel1.nm\_kolom2,**

**nm\_tabel2.nm\_kolom1, nm\_tabel2.nm\_kolom2**

**FROM tabel1**

**RIGHT JOIN tabel2**

**ON tabel1.nama\_kolom1(primary key)=tabel2.nama\_kolom1(foreign key yg mengacu ke tabel1)**

**WHERE kondisi;**

Contoh :

**select a.cust\_name,b.order\_date from customers a right join orders b on a.cust\_id=b.cust\_id**

1. SELF JOIN

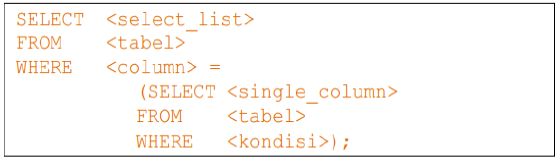
Self join adalah melakukan join dengan dirinya sendiri. Atau join dengan table yang sama. Sintak nya sbb :

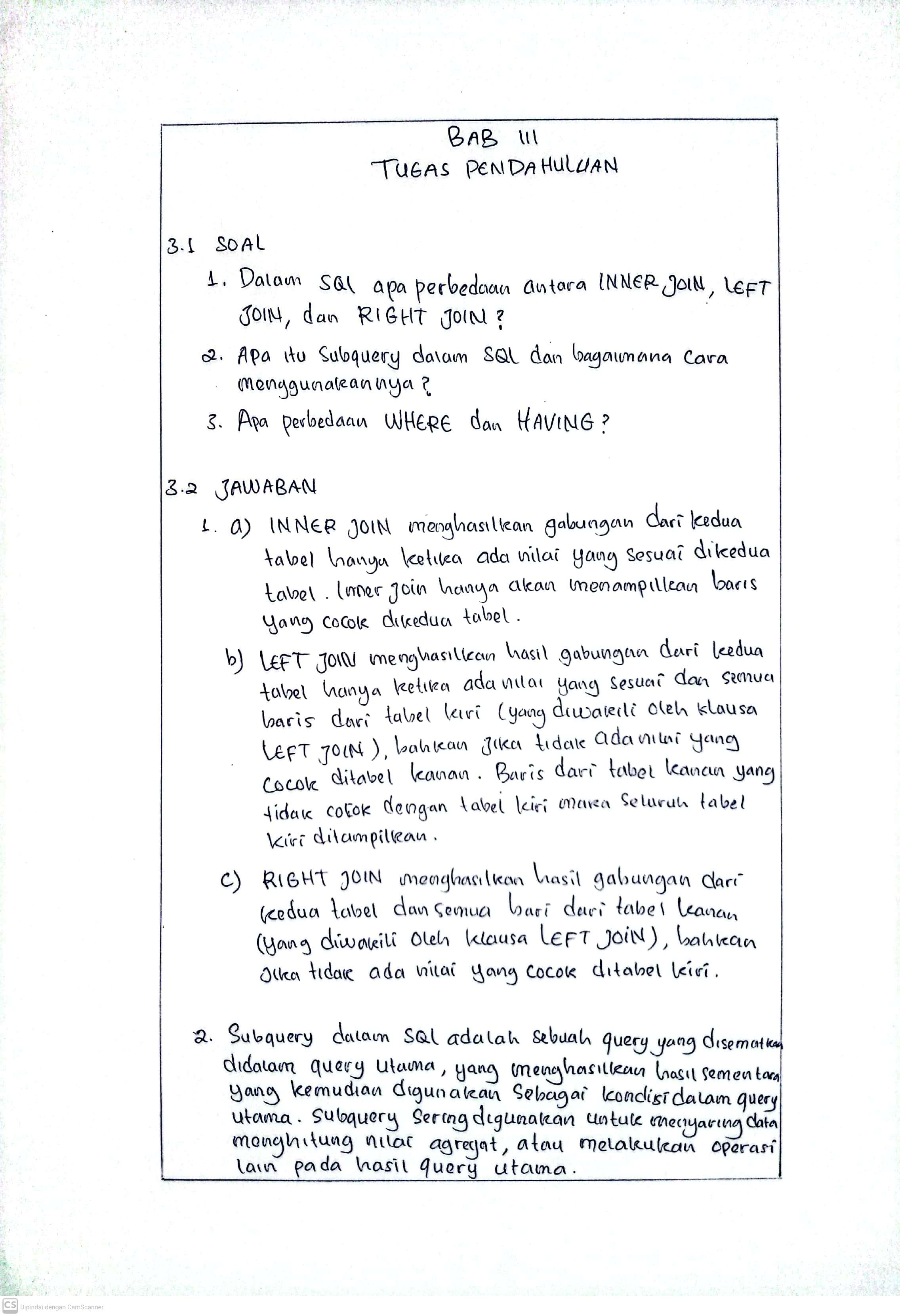
**select nama alias1\_table.kolom1, nama alias2\_table.kolom2, from table alias1 inner join table alias2 on alias1.kolom3=alias2.kolom3**

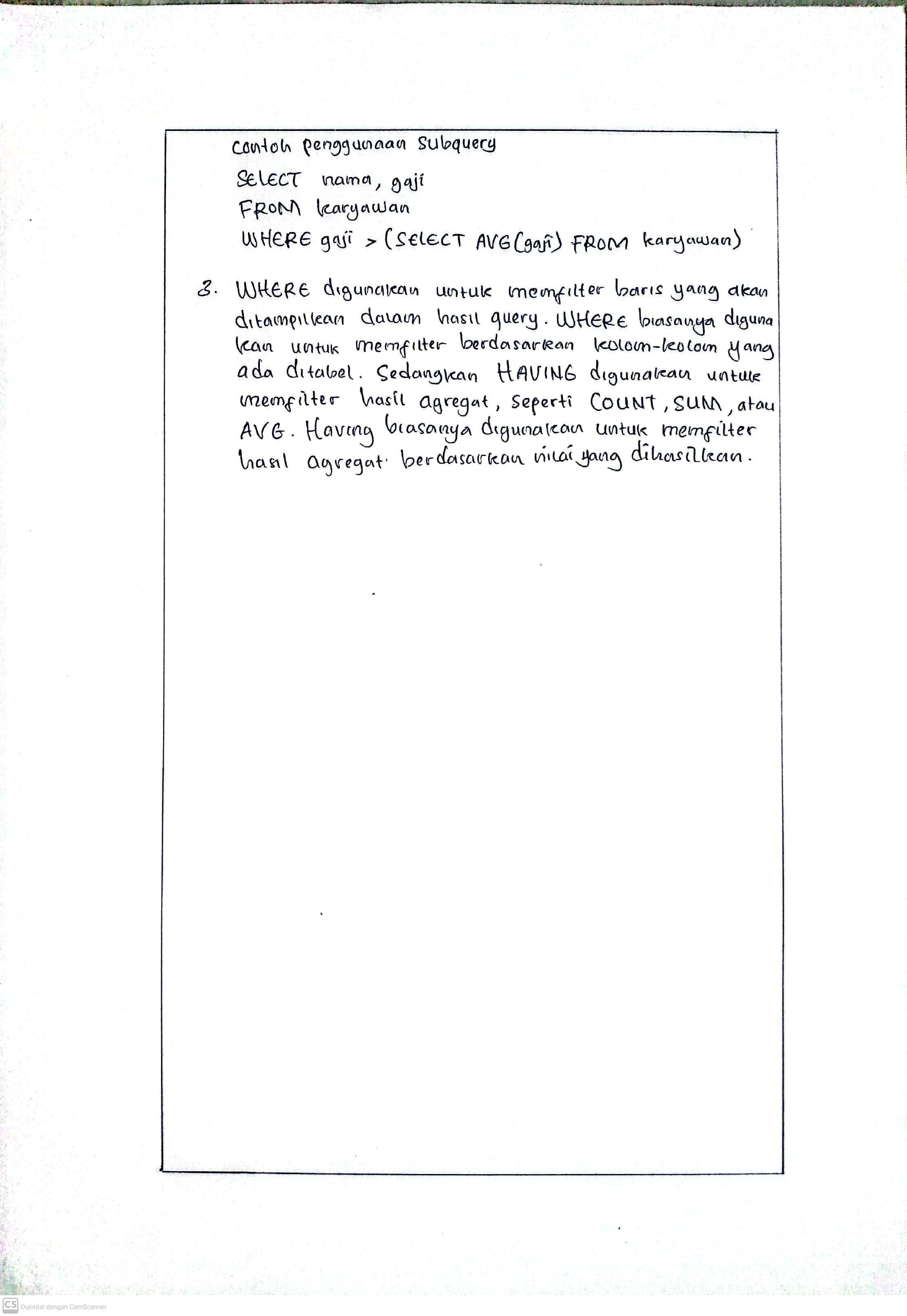
Contoh :

**select a.vend\_name,b.vend\_state, 'negaranya' ,b.vend\_country from vendors a inner join vendors b on a.vend\_id=b.vend\_id**

### Subquery

Subquery merupakan query didalam query. Umumnya, subquery ini dipakai untuk mencari data yang belum diketahui. Penggunaan query didalam query ini umumnya menjadi bagian dari kondisi. Sintak subquery adalah sebagai berikut :

****

****

**BAB IV**

**IMPLEMENTASI**

1. **Implementasi**
2. Buatlah sebuah database dengan nama coba !

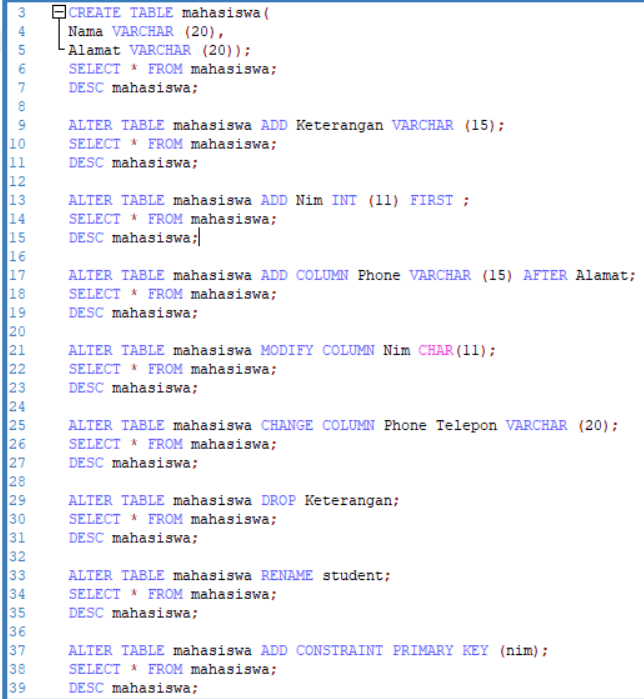
Query :



1. Buatlah tabel dengan nama mahasiswa didalam database coba, lalu

* Tambahkan sebuah kolom : keterangan (varchar 15), sebagai kolom terakhir !
* Tambahkan kolom nim (int 11) di awal (sebagai kolom pertama) !
* Sisipkan sebuah kolom dengan nama phone (varchar 15) setelah kolom alamat varchar (15) !
* Ubah atribut kolom nim menjadi char (11) !
* Ubah nama kolom phone menjadi telephone (varchar 20) !
* Hapus kolom keterangan dari tabel !
* Ganti nama tabel menjadi student !
* Jadikan nim sebagai primary key

Query :

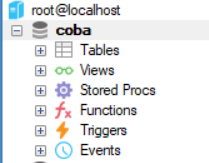


1. Screnshoot semua perintah-perintah SQL percobaan di atas beserta outputnya !

Query :



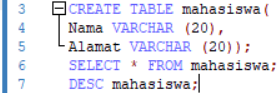
Output :



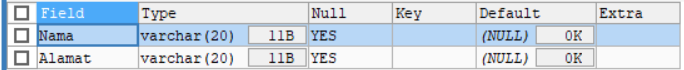
Penjelasan :

Pada query diatas saya membuat sebuah database baru dengan nama **coba** menggunakan **CREATE DATABASE** diikuti dengan nama database yaitu **coba.**

Query :



Output :



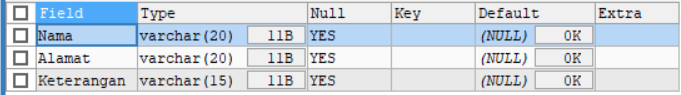
Penjelasan :

Pada query di atas saya membuat sebuah tabel baru dengan nama **mahasiswa** menggunakan **CREATE TABLE**, lalu saya membuat 2 kolom yaitu **Nama** dengan tipe data **varchar** dengan panjang karakter 20, lalu yang ke 2 yaitu **Alamat** dengan tipe data **varchar** dengan panjang karakter 20.

Query :



Output :



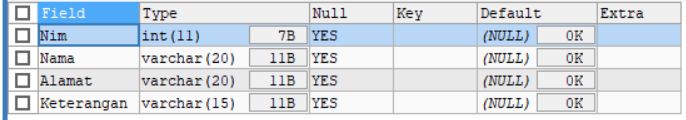
Penjelasan :

Pada query di atas saya menambahkan sebuah kolom di dalam tabel **mahasiswa** dengan menggunakan **ALTER TABLE mahasiswa (nama tabel) ADD Keterangan VARCHAR (15). ADD** berfungsi untuk menambahkan sesuatu, dilanjutkan dengan nama tabel yang ingin di tambahkan yaitu **Keterangan** dengan tipe data **varchar** dengan panjang karakter 15.

Query :



Ouput :



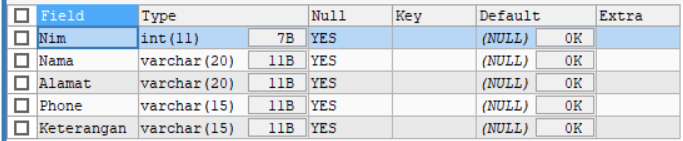
Penjelasan :

Pada query di atas saya menambahkan sebuah kolom di dalam tabel **mahasiswa** dengan menggunakan **ALTER TABLE mahasiswa (nama tabel) ADD Nim VARCHAR (15) FIRST.** Berbeda dengan sebelumnya pada query di atas terdapat query **FIRST** yang berfungsi untuk menempatkan kolom **Nim** berada di paling atas/pertama.

Query :



Output :



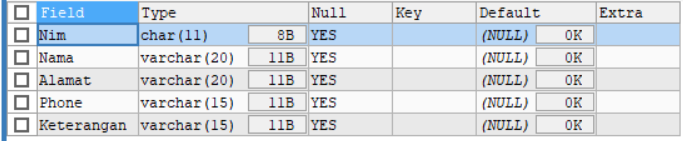
Penjelasan :

Pada query di atas saya menambahkan sebuah kolom di dalam tabel **mahasiswa** dengan menggunakan **ALTER TABLE mahasiswa (nama tabel) ADD COLUMN Phone VARCHAR (15) AFTER Alamat.** Berbeda dengan sebelumnya pada query di atas terdapat query **AFTER Alamat** yang berfungsi untuk menempatkan kolom **Phone** berada setelah kolom **Alamat**.

Query :



Output :



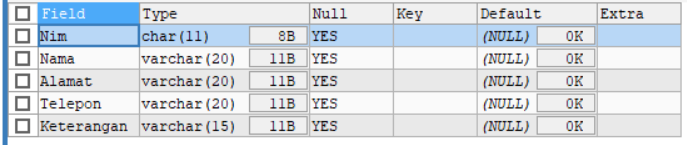
Penjelasan :

Pada query di atas saya mengubah sebuah atribut kolom **Nim** di dalam tabel **mahasiswa** dengan menggunakan **ALTER TABLE mahasiswa (nama tabel) MODIFY COLUMN Nim CHAR (11).** Jadi, atribut **Nim** yang awalnya **VARCHAR (15)** berubah menjadi **CHAR (11)**.

Query :



Output :



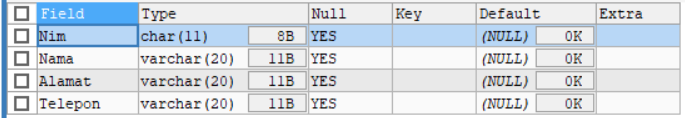
Penjelasan :

Pada query di atas saya mengubah sebuah nama kolom **Phone** dan atributnya di dalam tabel **mahasiswa** dengan menggunakan **ALTER TABLE mahasiswa (nama tabel) CHANGE COLUMN Phone Telepon VARCHAR (20).** Jadi, nama kolom **Phone** dengan atribut **VARCHAR (15)** berubah menjadi **Telepon VARCHAR (20)**.

Query :



Output :



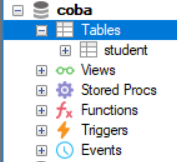
Penjelasan :

Pada query di atas saya menghapus sebuah nama kolom **Keterangan** di dalam tabel **mahasiswa** dengan menggunakan **ALTER TABLE mahasiswa (nama tabel) DROP Keterangan**. Query **DROP** tersebut berfungsi untuk menghapus kolom.

Query :



Output :



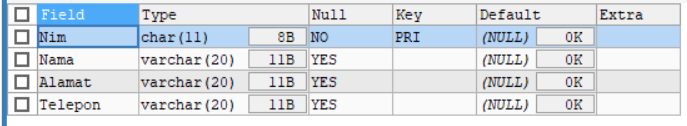
Penjelasan :

Pada query di atas saya mengubah nama tabel **mahasiswa** menjadi **student** dengan menggunakan **ALTER TABLE mahasiswa (nama tabel) RENAME student**.

Query :



Output :



Penjelasan :

Pada query di atas saya mengubah kolom **Nim** sebagai **PRIMARY KEY** dengan menggunakan **ALTER TABLE student (nama tabel) ADD CONSTRAINT PRIMARY KEY (nim)**.

1. Apa maksud dari int (11) ?

Maksud dari int (11) yaitu tipe data yang berupa angka bulat positif ataupun negatif dengan panjang karakter 11

1. Ketika kita melihat struktur tabel dengan perintah desc, ada kolom Null yang berisi Yes dan No. Apa maksudnya ?

* NULL (Yes), berfungsi ketika kita menginputkan sebuah data diinputkan ataupun tidak, MySql tidak akan eror jadi walaupun kita tidak menginputkan sebuah data tidak terjadi masalah. Berbeda dengan NOT NULLL (No).
* NOT NULL (No), pada Not Null kita d haruskan untuk menginputkan sebuah data, jika tidak d inputkan MySql akan terjadi eror.

**BAB V**

**PENUTUP**

1. **Analisa**

Dari hasil praktikum, praktikan menganalisa bahwa SQL adalah Structured Query Language yang berarti bahasa pemrograman standar yang memiliki kegunaan, fungsi dan tujuan utama untuk mempermudah proses pengelolaan database. Oleh sebab itu sangat penting bagi calon programmer untuk mempelajari materi pengenalan SQL terlebih dahulu sebelum mulai praktek coding.

Sama seperti semua bahasa pemrograman pada umumnya yang mempunyai kelebihan, SQL banyak digunakan karena sangat unggul dari sisi pengajuan request atau permintaan terhadap programnya. SQL memungkinkan untuk membuat berbagai permintaan tindakan terhadap database.

1. **Kesimpulan**

SQL (Structured Query Language) adalah sebuah bahasa yang digunakan untuk mengakses data dalam basis data relasional. Bahasa ini secara de facto merupakan bahasa standar yang digunakan dalam manajemen basis data relasional. Saat ini hampir semua server basis data yang ada mendukung bahasa ini untuk melakukan manajemen datanya.

Secara umum, SQL terdiri dari dua bahasa, yaitu Data Definition Language (DDL)dan Data Manipulation Language (DML). Implementasi DDL dan DML berbeda untuk tiap sistem manajemen basis data (SMBD),