LAPORAN RESMI

MODUL IV

SQL

NAMA	**:**	**ANISYAFAAH**
N.R.P	**:**	**220441100105**
DOSEN	**:**	**FITRI DAMAYANTI, S.Kom., M.Kom.**
ASISTEN	**:**	**AFFAN MAULANA ZULKARNAIN**
TGL PRAKTIKUM	**:**	**08 MEI 2024**

Disetujui : 14 Mei 2024
Asisten

AFFAN MAULANA ZULKARNAIN
20.04.411.00052

LABORATORIUM BISNIS INTELIJEN SISTEM
PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Di era modern yang dipenuhi dengan data, kebutuhan akan sistem manajemen basis data (DBMS) seperti SQL menjadi semakin penting. Pertama-tama, SQL menyediakan sebuah standar industri yang telah teruji dan terbukti efisien dalam mengelola data. Dengan SQL, pengguna dapat dengan mudah membuat, mengakses, dan mengelola basis data relasional, memungkinkan mereka untuk menyimpan dan mengambil informasi dengan cepat dan efisien. Selain itu, SQL juga menawarkan kemampuan untuk melakukan kueri kompleks, menghasilkan analisis yang mendalam dari data yang tersimpan.

Kedua, dalam lingkungan bisnis yang kompetitif, keamanan data menjadi sangat penting. SQL menyediakan beragam fitur keamanan yang dapat membantu melindungi data dari akses yang tidak sah. Dengan kontrol akses yang ketat, penggunaan SQL memastikan bahwa hanya pengguna yang berwenang yang memiliki akses ke data yang sensitif. Fitur seperti otentikasi, otorisasi, dan enkripsi data memastikan bahwa informasi penting tetap aman dan terlindungi dari ancaman keamanan eksternal maupun internal.

Terakhir, dengan perkembangan teknologi dan pertumbuhan data yang terus meningkat, skalabilitas menjadi faktor kunci dalam pemilihan sistem manajemen basis data. SQL memungkinkan perusahaan untuk mengelola volume data yang besar dengan mudah dan efisien. Dengan desain yang fleksibel, basis data SQL dapat diperluas sesuai dengan kebutuhan pertumbuhan perusahaan tanpa mengorbankan kinerja atau keamanan. Dengan demikian, SQL tetap menjadi pilihan utama dalam memenuhi kebutuhan manajemen data di era modern ini.

1.2 Tujuan

- Mampu memahami dan membuat perintah SQL lebuh dalam dengan berbagai kondisi, ekpresi, pengurutan
- Mampu menggabungkan beberapa argumen atau perintah pada beberapa tabel (join) dalam basis data

BAB II
DASAR TEORI

2.1 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
2. 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
3. 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.

2.1.1 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 6 komponen, antara lain:

a) SELECT. Diikuti oleh **<select_list>,** dapat berupa literal_value atau column_list atau asterisk (*).
b) 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.

c) WHERE. Diikuti oleh kondisi secara umum.
d) GROUP BY. Diikuti oleh **<select_list>.** Bagian ini muncul ketika ada fungsi- fungsi agregasi.
e) HAVING. Diikuti oleh kondisi hanya untuk fungsi -fungsi agregasi.
f) ORDER BY. Diikuti oleh <select_list>

Perintahnya :

```
SELECT     <select_list>
[FROM      <table_name>]
[WHERE     <kondisi1> [AND/OR <kondisi2>]]
[GROUP BY  <select_list>]
[HAVING    <kondisi1> [AND/OR <kondisi2>]]
[ORDER BY  <select_list>]
```

Keterangan :

- SELECT, INTO, FROM, WHERE, GROUP BY, HAVING DAN ORDER BY → kata kunci (keyword) yang harus disertakan jika kita membutuhkannya di dalam pengolahan data.
- select_list, table_source, search_condition, group_by_expression, order_expression → isian yang bisa kita ubah berdasarkan kebutuhan kita
- Kurung kotak [ ] → bagian tersebut boleh disertakan atau tidak, tergantung dari kebutuhan

2.1.2 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 :

Operator	Keterangan
=	Sama dengan
>	Lebih besar dari
>=	Lebih besar sama dengan
<	Kurang dari
<=	Kurang dari sama dengan
<> atau !=	Tidak sama dengan
BETWEEN .. AND ..	Diantara 2 nilai
IN (set)	Cocok dengan salah satu diantara daftar nilai
LIKE	Cocok dengan pola karakter
IS NULL	Sama dengan NULL

Perintahnya:

Select * From Nama_Table Where Nama_Field [Operator Relasional] Ketentuan;

Contoh:

Select * from customer where cust_id = ‘10003’ or cust_name =’wascals;

2.1.3 Mengurutkan Data (ASC, DESC, ORDER BY)

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;

2.1.4 Operator BETWEEN, IN, LIKE

1. **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';

2. **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');

3. **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.

2.1.5 Ekspresi Query

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

- **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;

- **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;

- **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;

2.1.6 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.

Fungsi	Keterangan
AVG	Menghitung rata-rata
COUNT	Menghitung cacah data /jumlah baris
MAX	Memperoleh nilai terbesar
MIN	Memperoleh nilai terkecil
SUM	Memperoleh jumlahan data

2.1.7 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.

```
SELECT  <select_list>
FROM    <table1>, <table2> [, ...]
WHERE   <table1.PK = table2.FK> [AND ...]
```

Perkembangan SQL ANSI

```
SELECT  <select_list>
FROM    <table1> JOIN <table2>
        ON < table1.PK = table2.FK> [[AND ...]
        JOIN ...];
```

Tipe Join

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

```
SELECT  <select_list>
FROM    <tabel1 sebagai kiri>
        <LEFT/RIGHT> [OUTER] JOIN
        <tabel2 sebagai kanan>
        ON <table1.PK = table2.FK> [AND ...];
```

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

a. 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;

b. 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;

c. 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:

1. LEFT OUTER JOIN atau biasa disebut left join
2. RIGHT OUTER JOIN atau biasa disebut righ join

a) 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

b) 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.

c) 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

2.1.8 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:

```
SELECT  <select_list>
FROM    <tabel>
WHERE   <column> =
           (SELECT <single_column>
           FROM    <tabel>
           WHERE   <kondisi>);
```

BAB III
TUGAS PENDAHULUAN

3.1 Soal

1. Bagaimana kalian menggunakan operasi JOIN dan ORDER BY dalam SQL untuk melakukan analisis data penjualan di beberapa cabang toko ? jelaskan bagaimana kalian akan menggabungkan data dari tabel (penjualan produk) dengan tabel (insomasi produk) menggunakan operasi JOIN, dan kemudian mengurutkan hasilnya berdasarkan kriteria tertentu, seperti total penjualan atau nama produk !
2. Jelaskan bagaimana kalian akan menggunakan fungsi agregasi seperti SUM, COUNT, AVG dalam SQL untuk menghitung total penjualan, jumlah produk yang terjual, dan rata-rata penjualan di setiap cabang toko. Sertakan cara kalian menggunakan operasi GROUP BY untuk mengelompokkan data penjualan berdasarkan kriteria tertentu, seperti cabang toko atau kategori produk. Selain itu, berikan contoh penggunaan operator spesifik seperti BETWEEN untuk memfilter data penjualan dalam rentang waktu tertentu !

3.2 Jawab

1. Untuk melakukan analisis data penjualan di beberapa cabang toko dapat menggunakan operasi JOIN dan ORDER BY. Operasi JOIN untuk menggabung antartabel dan operasi ORDER BY untuk mengurutkan hasil. Beberapa tahapan yang dapat dilakukan untuk menerapkannya yaitu :
   a) Operasi JOIN (Menggabungkan Tabel)
      Misalnya database toko memiliki tabel penjualan_produk (id penjualan, id produk, stok, total penjualan) dan tabel insomasi_produk (id produk, nama produk, harga). Maka cara menggabungkan kedua tabel tersebut yaitu :

```
SELECT p.*, ip.nama_produk, ip.harga
FROM penjualan_produk p
INNER JOIN insomasi_produk ip ON p.id_produk = ip.id_produk;
```

b) Operasi ORDER BY (Mengurutkan hasil)

Untuk mengurutkan hasil dari penggabungan kedua tabel sebelumnya berdasarkan kriteria tertentu (total penjualan) yaitu :

```
SELECT p.*, ip.nama_produk, ip.harga
FROM penjualan_produk p
INNER JOIN informasi_produk ip ON p.id_produk = ip.id_produk
ORDER BY p.total_penjualan ;
```

2. Terdapat beberapa tahap untuk menggunakan fungsi-fungsi tersebut, antara lain :

a) Penggunaan Agregasi dan ORDER BY

Untuk menghitung total penjualan, jumlah produk terjual, dan rata-rata dapat menggunakan agregasi, yaitu :

```
SELECT kategori_produk,
       SUM(total_penjualan) AS total_penjualan,
       COUNT(*) AS jumlah_produk_terjual,
       AVG(total_penjualan) AS rata_rata_penjualan
FROM penjualan_produk
GROUP BY kategori_produk ;
```

b) Penggunaan BETWEEN

Misalnya ingin memfilter data penjualan bulan Januari 2024, maka:

```
SELECT *
FROM penjualan_produk
WHERE tanggal_penjualan BETWEEN '2024-01-01' AND '2024-01-31';
```

BAB IV
IMPLEMENTASI

4.1 Source Code

4.1.1 Menampilkan Daftar Karyawan Beserta Nama Atasan

1. Mengisi Data Tabel Karyawan

a) Source Code

```
CREATE DATABASE Karyawan;

USE Karyawan;

CREATE TABLE Karyawan (
ID_Karyawan INT(11) NOT NULL PRIMARY KEY,
   Nama VARCHAR(100),
   Posisi VARCHAR(50),
   ID_Atasan INT(11)
);

INSERT INTO Karyawan (ID_Karyawan, Nama, Posisi, ID_Atasan)
VALUES
        (1, 'Budi Santoso', 'Manager', NULL),
        (2, 'Ani Wijaya', 'Supervisor', 1),
        (3, 'Cahyo Nugroho', 'Staf', NULL),
        (4, 'Dewi Kurniawan', 'Staf', 1),
        (5, 'Eka Setiawan', 'Staf', 1),
        (6, 'Fita Dewanti', 'Staf', 1),
        (7, 'Galih Susanto', 'Staf', 2),
        (8, 'Hani Maulana', 'Staf', 2),
        (9, 'Indra Wibowo', 'Staf', 2),
        (10, 'Joko Prasetyo', 'Staf', 2);
```

b) Penjelasan

Kode di atas digunakan untuk membuat database karyawan dan membuat tabel karyawan serta mengisinya dengan 10 data. Untuk membuat database karyawan menggunakan perintah CREATE DATABASE Karyawan. Untuk membuat tabel produk menggunakan perintah CREATE TABLE Karyawan

(namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id karyawan sebagai primary key, nama, posisi, dan id atasannya. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

2. Menampilkan Daftar Karyawan Beserta Nama Atasan

a) Source Code

```
-- Nomor 1
SELECT
B.Nama AS Nama_Karyawan, B.Posisi AS Posisi_Karyawan,
A.Nama AS Nama_Atasan, A.Posisi AS Posisi_Atasan
FROM Karyawan A
RIGHT JOIN Karyawan B ON A.ID_Karyawan = B.ID_Atasan
ORDER BY B.ID_Karyawan;
```

b) Penjelasan

Kode di atas digunakan untuk menampilkan nama karyawan beserta nama atasannya. Kode di atas menggunakan self join untuk menggabungkan tabel dengan dirinya sendiri dan menggunakan right join untuk menampilkan hasil berdasarkan tabel kanan. Jika hasil tidak ada yang cocok maka data akan otomatis terisi dengan NULL. Kemudian terdapat fungsi ORDER BY untuk mengurutkan data berdasarkan id karyawan dari tabel kanan.

4.1.2 Stored Procedure Rata – Rata Nilai Dari Setiap Mahasiswa

1. Mengisi Data Setiap Tabel

- **Mengisi Data Tabel Mahasiswa**

a) Source Code

```
CREATE DATABASE Mahasiswa;

USE Mahasiswa;

CREATE TABLE Mahasiswa (
```

```
 ID_Mahasiswa INT(11) NOT NULL PRIMARY KEY,
    Nama VARCHAR(100),
    Jurusan VARCHAR(50),
    Tanggal_Masuk DATE
);


INSERT INTO Mahasiswa (ID_Mahasiswa, Nama, Jurusan, Tanggal_Masuk) VALUES
(1, 'Anisah Nuril Izzati', 'Teknik Informatika', '2023-08-01'),
(2, 'Bayu Aditya Pratama', 'Manajemen', '2023-08-01'),
(3, 'Citra Wahyuni Putri', 'Akuntansi', '2023-08-01'),
(4, 'Dhika Putra Ramadhan', 'Teknik Sipil', '2023-08-01'),
(5, 'Eka Wahyu Kurniawan', 'Ilmu Komunikasi', '2023-08-01'),
(6, 'Fadhila Indah Permata', 'Sastra Inggris', '2023-08-01'),
(7, 'Gilang Fajar Saputra', 'Teknik Elektro', '2023-08-01'),
(8, 'Haniyah Amira Fitri', 'Kedokteran', '2023-08-01'),
(9, 'Ibrahim Hidayatullah', 'Hukum', '2023-08-01'),
(10, 'Joko Setiawan', 'Agribisnis', '2023-08-01');
```

b) Penjelasan

Kode di atas digunakan untuk membuat database mahasiswa dan membuat tabel mahasiswa serta mengisinya dengan 10 data. Untuk membuat database mahasiswa menggunakan perintah CREATE DATABASE Mahasiswa. Untuk membuat tabel mahasiswa menggunakan perintah CREATE TABLE Mahasiswa (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id mahasiswa sebagai primary key, nama, jurusan, dan tanggal masuk mahasiswa. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

- **Mengisi Data Tabel Mata Kuliah**

a) Source Code

```
CREATE TABLE Mata_Kuliah (
 ID_Matakuliah INT(11) NOT NULL PRIMARY KEY,
    Nama_Mata_Matakuliah VARCHAR(100),
    Sks INT(11),
```

```
   Semester VARCHAR(20)
);

INSERT INTO Mata_Kuliah (ID_Matakuliah,
Nama_Mata_Matakuliah, Sks, Semester) VALUES
(1, 'Pemrograman Dasar', 3, 'Genap'),
(2, 'Basis Data', 4, 'Genap'),
(3, 'Matematika Diskrit', 3, 'Genap'),
(4, 'Pengantar Teknologi Informasi', 2, 'Genap'),
(5, 'Statistika', 3, 'Genap'),
(6, 'Pemrograman Lanjut', 4, 'Genap'),
(7, 'Sistem Operasi', 3, 'Genap'),
(8, 'Manajemen Proyek TI', 3, 'Genap'),
(9, 'Jaringan Komputer', 4, 'Genap'),
(10, 'Pemrograman Web', 3, 'Genap');
```

b) Penjelasan

Kode di atas digunakan untuk membuat tabel mata kuliah serta mengisinya dengan 10 data. Untuk membuat tabel mata kuliah menggunakan perintah CREATE TABLE Mata_Kuliah (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id matakuliah sebagai primary key, nama mata kuliah, sks, dan semester. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

- **Mengisi Data Tabel Nilai**

a) Source Code

```
CREATE TABLE Nilai (
 ID INT(11) NOT NULL PRIMARY KEY,
   ID_Mahasiswa INT(11),
   ID_Matakuliah INT(11),
   Nilai VARCHAR(2),
   FOREIGN KEY (ID_Mahasiswa) REFERENCES Mahasiswa
(ID_Mahasiswa),
```

```
  FOREIGN KEY (ID_Matakuliah) REFERENCES Mata_Kuliah (ID_Matakuliah)
);

INSERT INTO Nilai (ID, ID_Mahasiswa, ID_Matakuliah, Nilai) VALUES
(1, 1, 1, 'A'),
(2, 1, 2, 'B'),
(3, 1, 3, 'B'),
(4, 2, 2, 'B+'),
(5, 3, 1, 'A'),
(6, 3, 2, 'B'),
(7, 3, 3, 'C'),
(8, 4, 4, 'B'),
(9, 5, 5, 'C+'),
(10, 6, 6, 'A'),
(11, 7, 7, 'A'),
(12, 8, 8, 'B'),
(13, 9, 9, 'A'),
(14, 10, 10, 'C');
```

b) Penjelasan

Kode di atas digunakan untuk membuat tabel nilai serta mengisinya dengan 10 data. Untuk membuat tabel nilai menggunakan perintah CREATE TABLE Nilai (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id sebagai primary key, id mahasiswa yang terhubung dengan tabel mahasiswa menggunakan FOREIGN KEY, id mata kuliah yang terhubung dengan tabel mata kuliah menggunakan FOREIGN KEY, dan nilai. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

2. Stored Procedure Rata – Rata Nilai Dari Setiap Mahasiswa

a) Source Code

```
-- Nomor 2
DELIMITER //
CREATE PROCEDURE Rata_Rata()
BEGIN
   SELECT Mahasiswa.ID_Mahasiswa, Mahasiswa.Nama AS 'Nama Mahasiswa', Mata_Kuliah.Semester AS 'Semester',
      AVG(
         CASE
            WHEN N.Nilai = 'A' THEN 4
            WHEN N.Nilai = 'B+' THEN 3.5
            WHEN N.Nilai = 'B' THEN 3
            WHEN N.Nilai = 'C+' THEN 2.5
            WHEN N.Nilai = 'C' THEN 2
            WHEN N.Nilai = 'D+' THEN 1.5
            WHEN N.Nilai = 'D' THEN 1
            ELSE NULL
         END
      ) AS 'Nilai Rata-Rata'
   FROM Mahasiswa
   JOIN Nilai N ON Mahasiswa.ID_Mahasiswa = N.ID_Mahasiswa
   LEFT JOIN Mata_Kuliah ON N.ID_Matakuliah = Mata_Kuliah.ID_Matakuliah
   GROUP BY Mahasiswa.Nama, Mata_Kuliah.Semester;
END //
DELIMITER ;

CALL Rata_Rata();
```

b) Penjelasan

Kode di atas digunakan untuk membuat prosedur dengan menampilkan rata – rata nilai setiap mahasiswa dari tabel mahasiswa dan nilai. Sintaks SQLnya terdapat beberapa kolom yang akan ditampilkan dan AVG untuk menghitung otomatis rata – rata dan juga fungsi CASE untuk sebuah kondisi dimana setiap nilai akan dikonversi menjadi sebuah angka. Tabel yang terlibat

terdapat tiga tabel sehingga perlu dihubungkan dengan fungsi JOIN. LEFT JOIN berguna untuk mencocokkan data dari tabel kanan namun jika terdapat data yayng tidak cocok maka akan otomatis diset NULL. GROUP BY berfungsi untuk mengurutkan data data selain data dalam agregasi.

4.2 Hasil

4.2.1 Menampilkan Daftar Karyawan Beserta Nama Atasan

1. Data Tabel Karyawan

ID_Karyawan	Nama	Posisi	ID_Atasan
1	Budi Santoso	Manager	NULL
2	Ani Wijaya	Supervisor	1
3	Cahyo Nugroho	Staf	NULL
4	Dewi Kurniawan	Staf	1
5	Eka Setiawan	Staf	1
6	Fita Dewanti	Staf	1
7	Galih Susanto	Staf	2
8	Hani Maulana	Staf	2
9	Indra Wibowo	Staf	2
10	Joko Prasetyo	Staf	2
NULL	NULL	NULL	NULL

2. Tampilan Daftar Karyawan Beserta Nama Atasan

Nama_Karyawan	Posisi_Karyawan	Nama_Atasan	Posisi_Atasan
Budi Santoso	Manager	NULL	NULL
Ani Wijaya	Supervisor	Budi Santoso	Manager
Cahyo Nugroho	Staf	NULL	NULL
Dewi Kurniawan	Staf	Budi Santoso	Manager
Eka Setiawan	Staf	Budi Santoso	Manager
Fita Dewanti	Staf	Budi Santoso	Manager
Galih Susanto	Staf	Ani Wijaya	Supervisor
Hani Maulana	Staf	Ani Wijaya	Supervisor
Indra Wibowo	Staf	Ani Wijaya	Supervisor
Joko Prasetyo	Staf	Ani Wijaya	Supervisor

4.2.2 Stored Procedure Rata – Rata Nilai Dari Setiap Mahasiswa

1. Hasil Data Setiap Tabel

- **Data Tabel Mahasiswa**

ID_Mahasiswa	Nama	Jurusan	Tanggal_Masuk
1	Anisah Nuril Izzati	Teknik Informatika	2023-08-01
2	Bayu Aditya Pratama	Manajemen	2023-08-01
3	Citra Wahyuni Putri	Akuntansi	2023-08-01
4	Dhika Putra Ramadhan	Teknik Sipil	2023-08-01
5	Eka Wahyu Kurniawan	Ilmu Komunikasi	2023-08-01
6	Fadhila Indah Permata	Sastra Inggris	2023-08-01
7	Gilang Fajar Saputra	Teknik Elektro	2023-08-01
8	Haniyah Amira Fitri	Kedokteran	2023-08-01
9	Ibrahim Hidayatullah	Hukum	2023-08-01
10	Joko Setiawan	Agribisnis	2023-08-01
NULL	NULL	NULL	NULL

- **Data Tabel Mata Kuliah**

ID_Matakuliah	Nama_Mata_Matakuliah	Sks	Semester
1	Pemrograman Dasar	3	Genap
2	Basis Data	4	Genap
3	Matematika Diskrit	3	Genap
4	Pengantar Teknologi Informasi	2	Genap
5	Statistika	3	Genap
6	Pemrograman Lanjut	4	Genap
7	Sistem Operasi	3	Genap
8	Manajemen Proyek TI	3	Genap
9	Jaringan Komputer	4	Genap
10	Pemrograman Web	3	Genap
NULL	NULL	NULL	NULL

- **Data Tabel Nilai**

ID	ID_Mahasiswa	ID_Matakuliah	Nilai
1	1	1	A
2	1	2	B
3	1	3	B
4	2	2	B+
5	3	1	A
6	3	2	B
7	3	3	C
8	4	4	B
9	5	5	C+
10	6	6	A
11	7	7	A
12	8	8	B
13	9	9	A
14	10	10	C
NULL	NULL	NULL	NULL

2. Stored Procedure Rata – Rata Nilai Dari Setiap Mahasiswa

ID_Mahasiswa	Nama Mahasiswa	Semester	Nilai Rata-Rata
1	Anisah Nuril Izzati	Genap	3.33333
2	Bayu Aditya Pratama	Genap	3.50000
3	Citra Wahyuni Putri	Genap	3.00000
4	Dhika Putra Ramadhan	Genap	3.00000
5	Eka Wahyu Kurniawan	Genap	2.50000
6	Fadhila Indah Permata	Genap	4.00000
7	Gilang Fajar Saputra	Genap	4.00000
8	Haniyah Amira Fitri	Genap	3.00000
9	Ibrahim Hidayatullah	Genap	4.00000
10	Joko Setiawan	Genap	2.00000

BAB V
PENUTUP

5.1 Analisa

Dari hasil praktikum, praktikan menganalisa bahwa penerapan SQL dalam sistem manajemen basis data (DBMS) memberikan beragam keuntungan bagi perusahaan dalam pengelolaan dan pengolahan data. Pertama, SQL menyediakan bahasa yang terstruktur dan mudah dipahami untuk berinteraksi dengan basis data relasional. Selain itu, SQL juga memfasilitasi kueri kompleks yang dapat menghasilkan informasi yang berharga dari data yang tersimpan, mendukung proses pengambilan keputusan yang lebih baik.

Kedua, penerapan SQL dalam sistem manajemen basis data memberikan kontrol yang kuat atas keamanan dan integritas data. SQL menyediakan fitur-fitur keamanan seperti otorisasi, autentikasi, dan enkripsi data yang membantu melindungi data dari akses yang tidak sah dan manipulasi yang tidak diizinkan. Dengan demikian, penerapan SQL dalam sistem manajemen basis data tidak hanya meningkatkan efisiensi operasional perusahaan tetapi juga meningkatkan keamanan dan integritas data, menjadikannya pilihan yang sangat diinginkan dalam lingkungan bisnis modern.

5.2 Kesimpulan

1. SQL diklasifikasikan menjadi tiga bagian, antara lain DDL (Data Definition Language), DML (Data Manipulation Language), dan DCL (Data Control Language)
2. SQL memiliki fungsi seperti SELECT, WHERE, ASC, DESC, ORDER BY, BETWEEN, IN, LIKE dan lain sebagainya untuk melakukan berbagai operasi pada data dalam basis data relasional.
3. SQL juga memiliki beberapa jenis fungsi JOIN seperti NATURAL JOIN, NNER JOIN, SELF JOIN dan lain sebagainya untuk menggabungkan data dari satu atau lebih tabel berdasarkan kondisi yang ditentukan.