**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**

## **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.

## **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**

## **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.

### **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:

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 :**



**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

### **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;

### **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;

### **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';

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.

* **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;

### **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 |

### **Multiple-table Query**

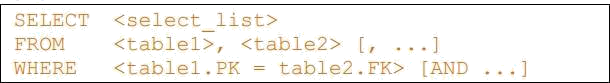
Data-data yang tersimpan dalam basis data, tersebar kedalam beberapa tabel. Tabel-tabel ini dihubungkan dengan yang namanya referential constraint, yaitu hubungan antara foreign key dan primary key.

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.

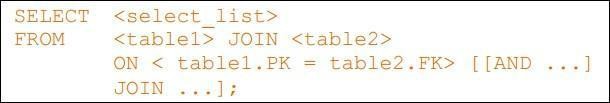
Perbedaannya sederhana, join menggunakan satu SELECT, sedangkan subquery menggunakan dua atau lebih SELECT (umumnya dikatakan sebagai SELECT within a SELECT).

**Join**

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

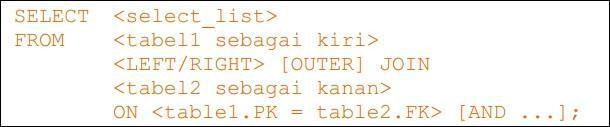


Perkembangan SQL ANSI



**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:

* 1. LEFT OUTER JOIN atau biasa disebut left join
  2. 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

1. **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.

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

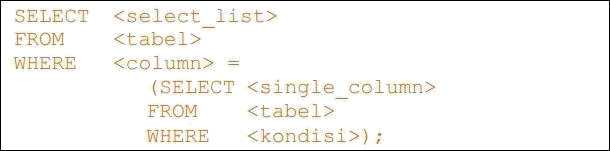
Contoh

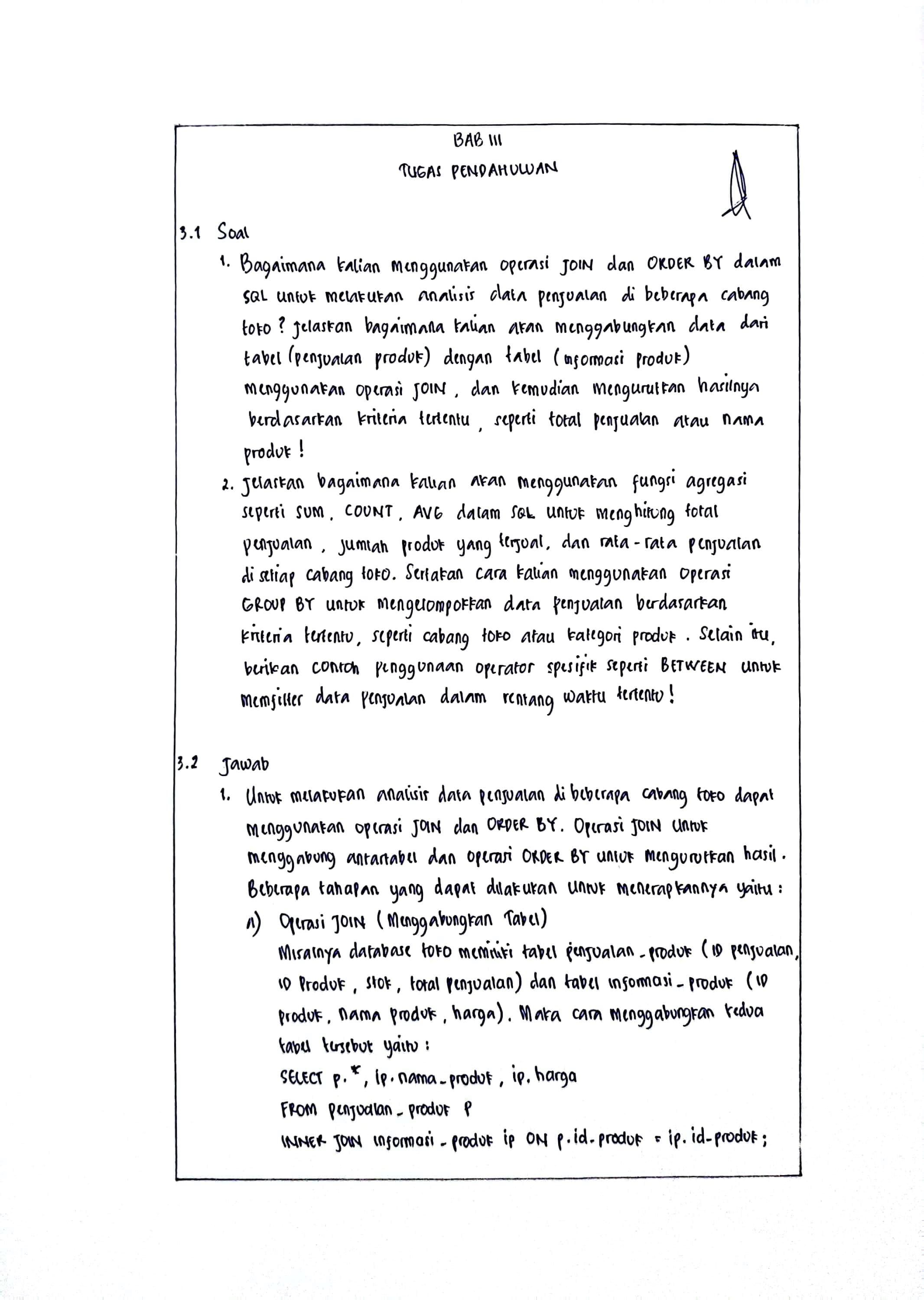
Select a.vend\_name,b.vend\_state, 'negaranya' ,b.vend\_country from vendors a inner join vendors b on a.vend\_id=b.vend\_id

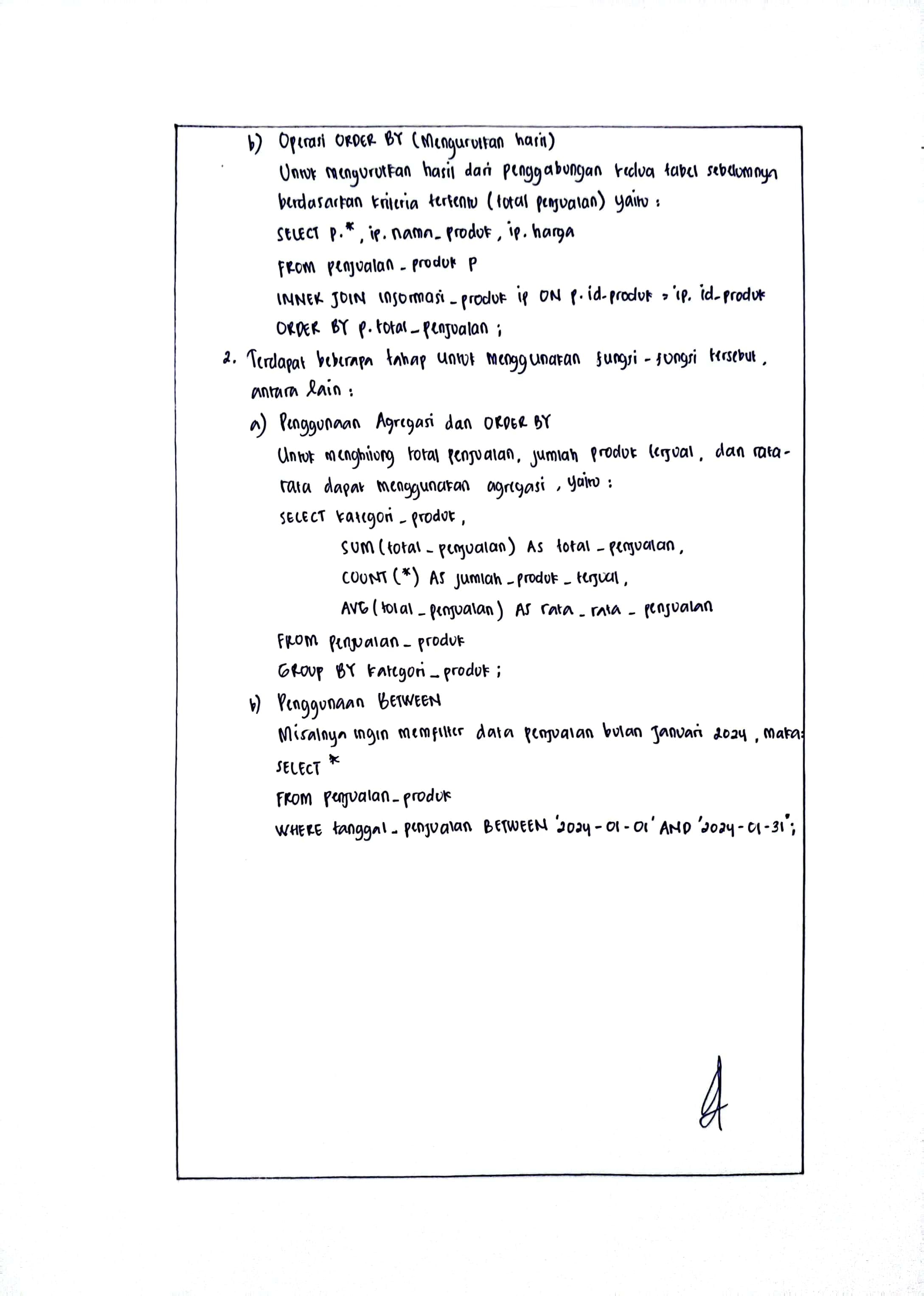
### **SubQuery**

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**

## **Source Code**

### **Menampilkan Daftar Karyawan Beserta Nama Atasan**

1. **Mengisi Data Tabel Karyawan**
2. **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); |

1. **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.

1. **Menampilkan Daftar Karyawan Beserta Nama Atasan**
2. **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; |

1. **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.

### **Stored Procedure Rata – Rata Nilai Dari Setiap Mahasiswa**

1. **Mengisi Data Setiap Tabel**

* **Mengisi Data Tabel Mahasiswa**

1. **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'); |

1. **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**

1. **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'); |

1. **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**

1. **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'); |

1. **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.

1. **Stored Procedure Rata – Rata Nilai Dari Setiap Mahasiswa**
2. **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(); |

1. **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.

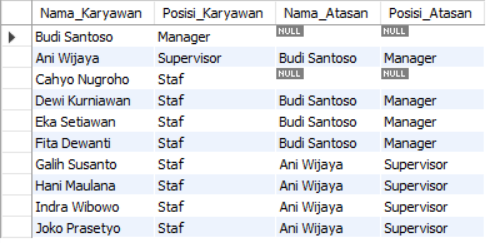
## **Hasil**

### **Menampilkan Daftar Karyawan Beserta Nama Atasan**

1. **Data Tabel Karyawan**

****

1. **Tampilan Daftar Karyawan Beserta Nama Atasan**

****

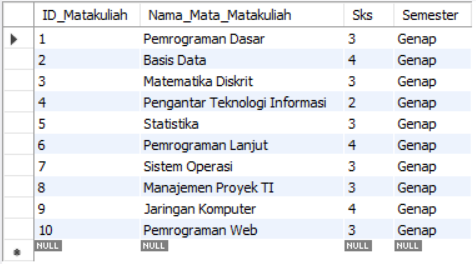
### **Stored Procedure Rata – Rata Nilai Dari Setiap Mahasiswa**

1. **Hasil Data Setiap Tabel**

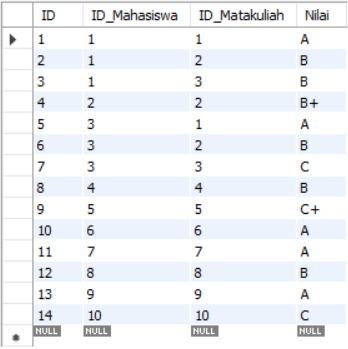
* **Data Tabel Mahasiswa**

****

* **Data Tabel Mata Kuliah**

****

* **Data Tabel Nilai**

****

1. **Stored Procedure Rata – Rata Nilai Dari Setiap Mahasiswa**

****

# **BAB V PENUTUP**

## **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.

## **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.