

PRAKTIKUM BASIS DATA LANJUT

LAPORAN RESMI MODUL 1 - 5



**DOSEN PENGAMPU : Budi Mukhamad Mulyo, S.Kom., M.T.
PARAF :**

DISUSUN OLEH :

Dyah Inkud Daifatur R.	23081010007
Regina Caeli Endyarni	23081010025
Sabila Ilma Irjayana	23081010039
Renita Enjel Siahaan	23081010147
Alvin Rama Saputra	23081010236
M. Wifaqul Azmi	23081010246

**LABORATORIUM PPSTI
PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PEMBANGUNAN "VETERAN" JAWA TIMUR
2024**

DAFTAR ISI

DAFTAR ISI	i
MODUL 1 : DATA DEFINITION LANGUAGE (DDL)	1
SOAL TUGAS PRAKTIKUM.....	1
ANALISIS MASALAH	1
HASIL PRAKTIKUM	1
PEMBAHASAN	9
KESIMPULAN	10
MODUL 2 : DATA MANIPULATION LANGUAGE (DML)	11
SOAL TUGAS PRAKTIKUM.....	11
ANALISIS MASALAH	11
HASIL PRAKTIKUM	12
PEMBAHASAN	17
KESIMPULAN	19
MODUL 3 : FUNGSI AGREGAT	20
SOAL TUGAS PRAKTIKUM.....	20
ANALISIS MASALAH	20
HASIL PRAKTIKUM	20
PEMBAHASAN	27
KESIMPULAN	32
MODUL 4 : ALJABAR RELASIONAL	33
SOAL TUGAS PRAKTIKUM.....	33
ANALISIS MASALAH	33
HASIL PRAKTIKUM	34
PEMBAHASAN	35
KESIMPULAN	43
MODUL 5 : QUERY BERSYARAT	44
SOAL TUGAS PRAKTIKUM.....	44
ANALISIS MASALAH	44
HASIL PRAKTIKUM	45
PEMBAHASAN	50
KESIMPULAN	50

MODUL 1

DATA DEFINITION LANGUAGE (DDL)

A. SOAL TUGAS PRAKTIKUM

1. Pelajari syntax Alter dan Rename dan implementasikan dalam setiap table dalam database.
2. ALTER TABLE.....ADD.....
ALTER TABLE.....DROP....
ALTER TABLE.....MODIFY....
RENAME TABLE.....TO.....
3. Tuliskan deskripsi singkat 4 syntax
(Deskripsi dari syntax ALTER ADD, ALTER DROP, ALTER MODIFY)
4. Screenshot hasil implementasi sebelum dan sesudah.

B. ANALISIS MASALAH

1. Bagaimana syntax atau cara membuat tabel database baru (Create) ?
2. Apa saja syntax - syntax dari Alter Table?
3. Bagaimana syntax dari Rename Table?
4. Apakah penulisan syntax dari Create, Alter, dan Rename telah dilakukan dengan benar selama percobaan?

C. HASIL PRAKTIKUM

Data Definition Language (DDL) merupakan sub perintah query SQL yang digunakan untuk mendefinisikan data di sebuah database dan table. Query yang digunakan dalam DDL dapat digunakan untuk membuat tabel baru, mengubah dataset, menghapus data, membuat indeks, menentukan struktur penyimpanan tabel dan sebagainya. Yang termasuk dalam Data Definition Language (DDL) adalah :

1. **CREATE**
CREATE merupakan sebuah perintah yang dapat digunakan ketika membuat sebuah database baru, baik itu berupa tabel baru atau sebuah kolom baru. Fungsi ini dapat digunakan untuk membuat sebuah query.
CREATE DATABASE nama_database
Contoh nya adalah sebagai berikut :

CREATE DATABASE dari Data Universitas

```
CREATE DATABASE DataUniversitas

CREATE TABLE Dept_emp (
  emp_no INT (11),
  dept_no CHAR (4),
  from_date DATE,
  to_date VARCHAR(1),
  CONSTRAINT pk_emp_no PRIMARY KEY (emp_no)
);

CREATE TABLE Prodi(
  dept_no CHAR (4),
  dept_name VARCHAR (40),
  CONSTRAINT pk_dept_no PRIMARY KEY (dept_no)
);

CREATE TABLE Kaprodi(
  dept_no CHAR (4),
  emp_no INT (11),
  from_date DATE,
  to_date DATE,
  CONSTRAINT pk_dept_no PRIMARY KEY (dept_no)
);

CREATE TABLE Jabatan(
  emp_no INT (11),
  title VARCHAR (50),
  from_date DATE,
  to_date TINYINT (1),
  CONSTRAINT pk_emp_no PRIMARY KEY (emp_no)
);

CREATE TABLE Pegawai(
  emp_no INT (11),
  birth_date DATE,
  first_name VARCHAR (14),
  last_name INT (15),
  gender ENUM ('M','F'),
  hire_date DATE,
  CONSTRAINT pk_emp_no PRIMARY KEY (emp_no)
);

DROP TABLE Gaji;

CREATE TABLE Gaji(
  emp_no INT (11),
  salary INT (11),
  from_date DATE,
  to_date DATE,
  CONSTRAINT pk_emp_no PRIMARY KEY (emp_no)
);
```

Hasil CREATE TABLE

dept_emp	gaji	jabatan	kaprodi	pegawai	prodi
emp_no dept_no from_date to_date	emp_no salary from_date to_date	emp_no title from_date to_date	dept_no emp_no from_date to_date	emp_no birth_date first_name last_name gender hire_date	dept_no dept_name

2. ALTER

ALTER Alter table adalah perintah dalam sistem operasi SQL yang berfungsi untuk mengubah struktur tabel yang sudah ada.

- a. Alter add adalah perintah SQL yang digunakan untuk menambahkan kolom baru ke dalam tabel basis data. Berikut syntaxnya :

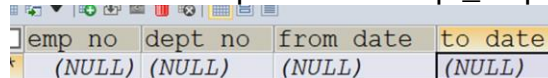
ALTER TABLE table_name ADD column_name datatype

Berikut adalah contoh ALTER ADD:

```
ALTER TABLE Dept_emp ADD COLUMN Address INT;  
ALTER TABLE Prodi ADD COLUMN Major INT;  
ALTER TABLE Kaprodi ADD COLUMN Nama INT;  
ALTER TABLE Jabatan ADD COLUMN Ruangan INT;  
ALTER TABLE Pegawai ADD COLUMN Age INT;  
ALTER TABLE Gaji ADD COLUMN Potongan INT;
```

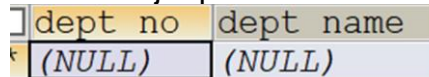
Sebelum :

- Kolom Address pada tabel Dept_emp



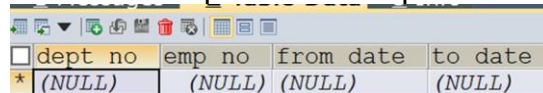
emp no	dept no	from date	to date
(NULL)	(NULL)	(NULL)	(NULL)

- Kolom Major pada tabel Prodi



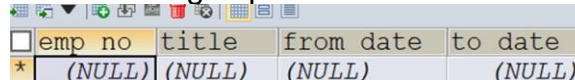
dept no	dept name
(NULL)	(NULL)

- Kolom Nama pada tabel Kaprodi



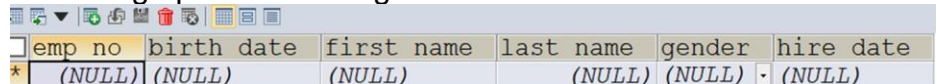
dept no	emp no	from date	to date
(NULL)	(NULL)	(NULL)	(NULL)

- Kolom Ruangan pada tabel Jabatan



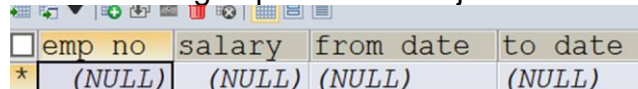
emp no	title	from date	to date
(NULL)	(NULL)	(NULL)	(NULL)

- Kolom Age pada tabel Pegawai



emp no	birth date	first name	last name	gender	hire date
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

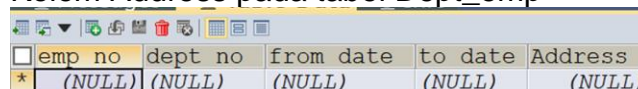
- Kolom Potongan pada tabel Gaji



emp no	salary	from date	to date
(NULL)	(NULL)	(NULL)	(NULL)

Sesudah :

- Kolom Address pada tabel Dept_emp



emp no	dept no	from date	to date	Address
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

- Kolom Major pada tabel Prodi

dept no	dept name	Major
(NULL)	(NULL)	(NULL)

- Kolom Nama pada tabel Kaprodi

dept no	emp no	from date	to date	Nama
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

- Kolom Ruangan pada tabel Jabatan

emp no	title	from date	to date	Ruangan
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

- Kolom Age pada tabel Pegawai

emp no	birth date	first name	last name	gender	hire date	Age
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

- Kolom Potongan pada tabel Gaji

emp no	salary	from date	to date	Potongan
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

- b. Alter Drop adalah perintah yang digunakan untuk menghapus kolom, constraint, atau objek lainnya dari tabel. Berikut syntaxnya :

ALTER TABLE table_name Drop column_name

Berikut ini adalah contoh dari Alter Drop :

```
ALTER TABLE Dept_emp DROP COLUMN Address;
ALTER TABLE Prodi DROP COLUMN Major;
ALTER TABLE Kaprodi DROP COLUMN Nama;
ALTER TABLE Jabatan DROP COLUMN Ruangan;
ALTER TABLE Pegawai DROP COLUMN Age;
ALTER TABLE Gaji DROP COLUMN Potongan;
```

Sebelum :

- Kolom Address dari tabel Dept_emp

emp no	dept no	from date	to date	Address
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

- Kolom Major dari tabel Prodi

dept no	dept name	Major
(NULL)	(NULL)	(NULL)

- Kolom Nama dari tabel Kaprodi

dept no	emp no	from date	to date	Nama
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

- Kolom Ruangan dari tabel Jabatan

emp no	title	from date	to date	Ruangan
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

- Kolom Age dari tabel Pegawai

emp no	birth date	first name	last name	gender	hire date	Age
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

- Kolom Potongan dari tabel Gaji

emp no	salary	from date	to date	Potongan
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

Sesudah :

- Kolom Address dari tabel Dept_emp

emp no	dept no	from date	to date
(NULL)	(NULL)	(NULL)	(NULL)

- Kolom Major dari tabel Prodi

dept no	dept name
(NULL)	(NULL)

- Kolom Nama dari tabel Kaprodi

dept no	emp no	from date	to date
(NULL)	(NULL)	(NULL)	(NULL)

- Kolom Ruangan dari tabel Jabatan

emp no	title	from date	to date
(NULL)	(NULL)	(NULL)	(NULL)

- Kolom Age dari tabel Pegawai

emp no	birth date	first name	last name	gender	hire date
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

- Kolom Potongan dari tabel Gaji

emp no	salary	from date	to date
(NULL)	(NULL)	(NULL)	(NULL)

- c. Alter Modify adalah perintah dalam sistem operasi SQL yang digunakan untuk memodifikasi kolom pada tabel yang sudah ada. Berikut syntaxnya :

ALTER TABLE table_name MODIFY column_name datatype

Berikut ini adalah contoh implementasi Alter Modify :

```
ALTER TABLE Dept_emp MODIFY COLUMN to_date DATE;
ALTER TABLE Prodi MODIFY COLUMN dept_name CHAR;
ALTER TABLE Kaprodi MODIFY COLUMN dept_no INT;
ALTER TABLE Jabatan MODIFY COLUMN to_date DATE;
ALTER TABLE Pegawai MODIFY COLUMN emp_no INT;
ALTER TABLE Gaji MODIFY COLUMN emp_no CHAR;
```

Sebelum :

- Kolom to_date dari tabel Dept_emp

```
/*Column Information*/
-----
```

Field	Type	Collation	Null
emp_no	int(11)	(NULL)	NO
dept_no	char(4)	utf8mb4_general_ci	YES
from_date	date	(NULL)	YES
to_date	varchar(1)	utf8mb4_general_ci	YES

- Kolom dept_name dari tabel Prodi

```
/*Column Information*/
-----
```

Field	Type	Collation	Null
dept_no	char(4)	utf8mb4_general_ci	NO
dept_name	varchar(40)	utf8mb4_general_ci	YES

- Kolom dept_no dari tabel Kaprodi

```
/*Column Information*/
-----
```

Field	Type	Collation	Null
dept_no	char(4)	utf8mb4_general_ci	NO
emp_no	int(11)	(NULL)	YES
from_date	date	(NULL)	YES
to_date	date	(NULL)	YES

- Kolom to_date dari tabel Jabatan

```
/*Column Information*/
-----
```

Field	Type	Collation	Null
emp_no	int(11)	(NULL)	NO
title	varchar(50)	utf8mb4_general_ci	YES
from_date	date	(NULL)	YES
to_date	tinyint(1)	(NULL)	YES

- Kolom emp_no dari tabel Pegawai

```
/*Column Information*/
-----
```

Field	Type	Collation	Null
emp_no	char(1)	utf8mb4_general_ci	NO
birth_date	date	(NULL)	YES
first_name	varchar(14)	utf8mb4_general_ci	YES
last_name	int(15)	(NULL)	YES
gender	enum('M','F')	utf8mb4_general_ci	YES
hire_date	date	(NULL)	YES

- Kolom emp_no dari tabel Gaji

```
/*Column Information*/
-----
```

Field	Type	Collation	Null
emp_no	int(11)	(NULL)	NO
salary	int(11)	(NULL)	YES
from_date	date	(NULL)	YES
to_date	date	(NULL)	YES

Sesudah :

- Kolom to_date dari tabel Dept_emp

```
/*Column Information*/
-----
```

Field	Type	Collation	Null
emp_no	int(11)	(NULL)	NO
dept_no	char(4)	utf8mb4_general_ci	YES
from_date	date	(NULL)	YES
to_date	date	(NULL)	YES

- Kolom dept_name dari tabel Prodi

```
/*Column Information*/
-----
```

Field	Type	Collation	Null
dept_no	char(4)	utf8mb4_general_ci	NO
dept_name	char(1)	utf8mb4_general_ci	YES

- Kolom dept_no dari tabel Kaprodi

```
/*Column Information*/
-----
```

Field	Type	Collation	Null
dept_no	int(11)	(NULL)	NO
emp_no	int(11)	(NULL)	YES
from_date	date	(NULL)	YES
to_date	date	(NULL)	YES

- Kolom to_date dari tabel Jabatan

```
/*Column Information*/
-----
```

Field	Type	Collation	Null
emp_no	int(11)	(NULL)	NO
title	varchar(50)	utf8mb4_general_ci	YES
from_date	date	(NULL)	YES
to_date	date	(NULL)	YES

- Kolom emp_no dari tabel Pegawai

```
/*Column Information*/
-----
```

Field	Type	Collation	Null
emp_no	int(11)	(NULL)	NO
birth_date	date	(NULL)	YES
first_name	varchar(14)	utf8mb4_general_ci	YES
last_name	int(15)	(NULL)	YES
gender	enum('M','F')	utf8mb4_general_ci	YES
hire_date	date	(NULL)	YES

- Kolom emp_no dari tabel Gaji

```
/*Column Information*/
-----
```

Field	Type	Collation	Null
emp_no	char(1)	utf8mb4_general_ci	NO
salary	int(11)	(NULL)	YES
from_date	date	(NULL)	YES
to_date	date	(NULL)	YES

3. RENAME

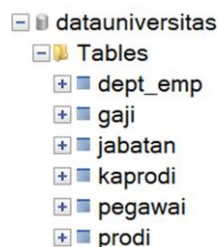
RENAME TABLE adalah pernyataan yang digunakan untuk mengganti nama tabel yang ada di dalam skema tertentu. Pernyataan ini dapat digunakan sebagai alternatif dari pernyataan ALTER TABLE saat ingin menata ulang skema tabel yang sudah ada.

RENAME TABLE old_name TO new_name

Contoh RENAME TABLE sebagai berikut :

```
RENAME TABLE Dept_emp TO Faculty;
RENAME TABLE Pegawai TO Employee;
RENAME TABLE Gaji TO Salary;
RENAME TABLE Jabatan TO Pangkat;
RENAME TABLE Kaprodi TO KoorProdi;
RENAME TABLE Prodi TO ProgramStudi;
```

Sebelum :



A screenshot of a database management tool showing the schema for 'datauniversitas'. Under the 'Tables' section, the following tables are listed: dept_emp, gaji, jabatan, kaprodi, pegawai, and prodi. Each table has a small icon to its left.

Sesudah :



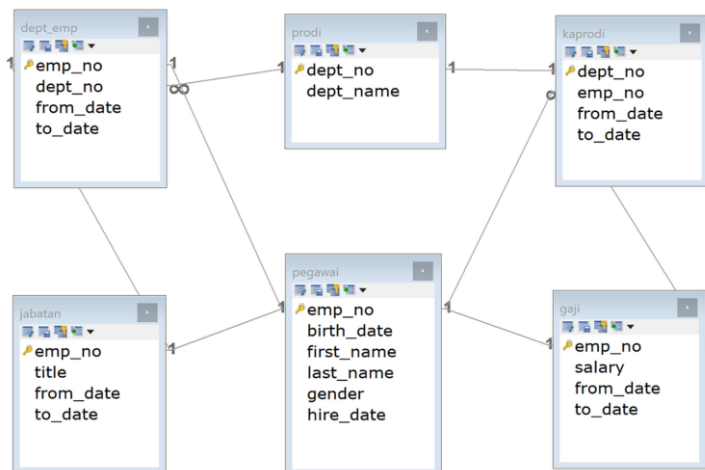
A screenshot of a database management tool showing the schema for 'datauniversitas' after the RENAME operation. Under the 'Tables' section, the following tables are listed: employee, faculty, koorprodi, pangkat, programstudi, and salary. Each table has a small icon to its left.

4. FOREIGN KEY

Menambahkan foreign key ke dalam query dengan syntax sebagai berikut :

```
ALTER TABLE Dept_emp ADD CONSTRAINT 'emp_no pegawai' FOREIGN KEY ('emp_no') REFERENCES 'Jabatan'('emp_no');
ALTER TABLE Dept_emp ADD CONSTRAINT 'emp_jabatan' FOREIGN KEY ('emp_no') REFERENCES 'Pegawai'('emp_no');
ALTER TABLE Dept_emp ADD CONSTRAINT 'dept_no prodi' FOREIGN KEY ('dept_no') REFERENCES 'Prodi'('dept_no');
ALTER TABLE Prodi ADD CONSTRAINT 'prodi_dept' FOREIGN KEY ('dept_no') REFERENCES 'Dept_emp'('dept_no');
ALTER TABLE Prodi ADD CONSTRAINT 'prodi_kaprodi' FOREIGN KEY ('dept_no') REFERENCES 'Kaprodi'('dept_no');
ALTER TABLE Kaprodi ADD CONSTRAINT 'kaprodi_prodi' FOREIGN KEY ('dept_no') REFERENCES 'Prodi'('dept_no');
ALTER TABLE Kaprodi ADD CONSTRAINT 'kaprodi_pegawai' FOREIGN KEY ('emp_no') REFERENCES 'Pegawai'('emp_no');
ALTER TABLE Kaprodi ADD CONSTRAINT 'kaprodi_gaji' FOREIGN KEY ('emp_no') REFERENCES 'Gaji'('emp_no');
ALTER TABLE Jabatan ADD CONSTRAINT 'jabatan_pegawai' FOREIGN KEY ('emp_no') REFERENCES 'Pegawai'('emp_no');
ALTER TABLE Jabatan ADD CONSTRAINT 'jabatan_dept' FOREIGN KEY ('emp_no') REFERENCES 'Dept_emp'('emp_no');
ALTER TABLE Pegawai ADD CONSTRAINT 'gaji_pegawai' FOREIGN KEY ('emp_no') REFERENCES 'Gaji'('emp_no');
ALTER TABLE Pegawai ADD CONSTRAINT 'pegawai_kaprodi' FOREIGN KEY ('emp_no') REFERENCES 'Kaprodi'('emp_no');
ALTER TABLE Gaji ADD CONSTRAINT 'pegawai_gaji' FOREIGN KEY ('emp_no') REFERENCES 'Kaprodi'('emp_no');
ALTER TABLE Gaji ADD CONSTRAINT 'gaji_emp_no' FOREIGN KEY ('emp_no') REFERENCES 'Pegawai'('emp_no');
```

Hasil dari penambahan foreign key pada query :



D. PEMBAHASAN

Pada Syntax Create, Alter, dan Rename yang telah dicoba diatas terdapat beberapa kekurangan yang mungkin sering terjadi dan berikut juga solusi dari masalah tersebut :

- **CREATE** : Kekurangan yang sering terjadi pada syntax Create yakni penggunaan tipe data yang tidak sesuai dengan kebutuhannya. Maka dari itu, sebelum membuat tabel perlu untuk menganalisis tipe data yang sesuai berdasarkan volume data dan kebutuhan operasionalnya.
- **ALTER ADD** : Kekurangan dari Alter Add yang sering terjadi yakni tidak adanya default value pada data yang sudah ada di tabel. Maka, kita perlu menentukan default value atau memperbarui baris data setelah penambahan kolom.
- **ALTER DROP** : Pada Alter Drop kita akan kehilangan data jika kolom tersebut dihapus, jadi kita perlu berhati-hati dalam penggunaan alter drop ini, atau juga kita dapat membackup data sebelum menghapus kolom dan pastikan bahwa kolom tersebut tidak lagi digunakan.
- **ALTER MODIFY** : Kekurangan pada Alter Modify yang umum terjadi yakni adanya ketidakcocokan dengan tipe data baru. Solusinya yakni, sebelum

kuta memodifikasi kolom, pastikan bahwa data yang ada sesuai dengan tipe data baru yang diinginkan.

- **RENAME** : Kekurangannya yakni, nama yang baru mungkin akan mempengaruhi integritas referensial jika tabel lain bergantung pada nama tabel atau kolom lama. Perlu melakukan pengecekan relasi antar tabel sebelum melakukan perubahan nama dan memastikan untuk memperbarui referensi yang di perlukan.

E. KESIMPULAN

Pada Operasi Create, Alter Add, Alter Drop, Alter Modify, serta Rename merupakan dasar dari manipulasi struktur tabel dalam database yang memiliki tantangan tersendiri. Adanya kekurangan atau kesalahan pada percobaan syntax SQL tersebut disebabkan karena human error yakni kurangnya pemahaman terhadap perintah SQL, serta kesalahan logika yang berdampak pada hilangnya data atau ketidakcocokan tipe data. Oleh karena itu, penting untuk melakukan analisis mendalam sebelum melakukan implementasi, serta menerapkan solusi yang tepat untuk mengatasi kekurangan yang ditemukan selama praktikum.

MODUL 2

DATA MANIPULATION LANGUAGE (DML)

A. SOAL TUGAS PRAKTIKUM

Soal sebelumnya dan perubahannya:

```
-- 1. Tampilkan informasi buku yang diterbitkan Erlangga
-- Perubahan Soal : Tampilkan informasi pegawai yang ada di sebuah PT X

-- 2. Tampilkan judul buku yang termasuk dalam kategori komputer.
-- Perubahan Soal : Tampilkan nama petugas dengan gender laki-laki

-- 3. Tampilkan nama anggota yg meminjam lebih dari 2 buku
-- Perubahan Soal : Tampilkan nama petugas yang lahir mulai tahun 1960

-- 4. tampilkan nama anggota yang terlambat mengembalikan buku
-- Perubahan Soal : Tampilkan nama pegawai laki-laki yang lahir di tahun 1960 yang di hire ditahun 1995.

-- 5 Tampilkan nama anggota, nama jurusan yang mahasiswanya terlambat mengembalikan buku
-- Perubahan Soal : Tampilkan nama dan jenis kelamin petugas yang mulai bekerja sebelum tahun 1986

-- 6. Tampilkan nama anggota dan judul buku yang terlambat dikembalikan
-- Perubahan Soal : Tampilkan nama petugas dan tanggal bekerja (hire date) yang di hire atau bekerja diatas tahun 1990
```

B. ANALISIS MASALAH

1. "Tampilkan informasi pegawai yang ada di sebuah PT X"
 - Analisis permasalahan: Pertanyaan ini meminta informasi tentang pegawai yang bekerja di PT X.
 - Pertanyaan: Bagaimana cara menggunakan query SELECT pada tabel yang menyimpan data pegawai, dengan filter berdasarkan syarat tertentu (WHERE clause)?
2. "Tampilkan nama petugas dengan gender laki-laki"
 - Analisis permasalahan: Pertanyaan ini meminta informasi tentang nama petugas dengan gender laki-laki.
 - Pertanyaan: Bagaimana cara menggunakan query SELECT pada tabel yang menyimpan data petugas, dengan filter berdasarkan gender?
3. "Tampilkan nama petugas yang lahir mulai tahun 1960"
 - Analisis permasalahan: Pertanyaan ini meminta informasi tentang nama petugas yang lahir mulai tahun 1960.
 - Pertanyaan: Bagaimana cara menggunakan query SELECT pada tabel yang menyimpan data petugas, dengan filter berdasarkan tahun lahir?




4. "Tampilkan nama pegawai laki-laki yang lahir di tahun 1960 yang di*hire* ditahun 1995".
 - Analisis permasalahan: Pertanyaan ini meminta informasi tentang pegawai laki-laki yang lahir di tahun 1960 dan di*hire* tahun 1995.
 - Pertanyaan: Bagaimana cara menggunakan query SELECT pada tabel yang menyimpan data pegawai, dengan filter berdasarkan jenis kelamin, tahun lahir, dan tahun di*hire*?
5. "Tampilkan nama dan jenis kelamin petugas yang mulai bekerja sebelum tahun 1986".
 - Analisis permasalahan: Pertanyaan ini meminta informasi tentang petugas yang mulai bekerja sebelum tahun 1986, termasuk nama dan jenis kelamin mereka.
 - Pertanyaan: Bagaimana cara menggunakan query SELECT pada tabel yang menyimpan data petugas, dengan filter berdasarkan tahun mulai bekerja?
6. "Tampilkan nama petugas dan tanggal bekerja (*hire date*) yang di*hire* atau bekerja diatas tahun 1990".
 - Analisis permasalahan: Pertanyaan ini meminta informasi tentang petugas yang di*hire* atau mulai bekerja di atas tahun 1990, termasuk nama dan tanggal bekerja.
 - Pertanyaan: Bagaimana cara menggunakan query SELECT pada tabel yang menyimpan data petugas, dengan filter berdasarkan tahun di*hire* atau mulai bekerja?
7. Bagaimana fungsi-fungsi lain selain select yang ada pada DML?







C. HASIL PRAKTIKUM

Soal 1 dan Query.

```
120      -- 1. Tampilkan informasi buku yang diterbitkan Erlangga
121      -- Perubahan Soal : Tampilkan informasi pegawai yang ada di sebuah PT X
122
123 •     SELECT * FROM employee;
```


Hasil *run* query:

Result Grid	 Filter Rows:	 Edit:	 Export:			
	emp_no	birth_date	first_name	last_name	gender	hire_date
▶	10001	1953-09-02	Georgi	Facello	M	1986-06-26
	10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
	10003	1959-12-03	Parto	Bamford	M	1986-08-28
	10004	1954-05-01	Christian	Koblick	M	1986-12-01
	10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
	10006	1953-04-20	Anneke	Preusig	F	1989-06-02
	10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
	10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
	10009	1952-04-19	Sumant	Peac	F	1985-02-18
	10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24
	10011	1953-11-07	Mary	Sluis	F	1990-01-22
	10012	1960-10-04	Patricio	Bridgland	M	1992-12-18
	10013	1963-06-07	Eberhardt	Terkki	M	1985-10-20
	10014	1956-02-12	Berni	Genin	M	1987-03-11
	10015	1959-08-19	Guoxiang	Nooteboom	M	1987-07-02
	10016	1961-05-02	Kazuhiro	Cappelletti	M	1995-01-27
	10017	1958-07-06	Cristinel	Bouloucos	F	1993-08-03
	10018	1954-06-19	Kazuhide	Peha	F	1987-04-03
	10019	1953-01-23	Lillian	Haddadi	M	1999-04-30
	10020	1952-12-24	Mayuko	Warwick	M	1991-01-26
	10021	1960-02-20	Ramzi	Erde	M	1988-02-10
	10022	1952-07-08	Shahaf	Famili	M	1995-08-22
	10023	1953-09-29	Bojan	Montema...	F	1989-12-17
	10024	1958-09-05	Suzette	Petty	F	1997-05-19
	10025	1958-10-31	Prasadram	Heyers	M	1987-08-17

Result Grid	  Filter Rows:	 Edit:		   Exp		
	emp_no	birth_date	first_name	last_name	gender	hire_date
	10026	1953-04-03	Yongqiao	Bertziss	M	1995-03-20
	10027	1962-07-10	Divier	Reistad	F	1989-07-07
	10028	1963-11-26	Domenick	Tempesti	M	1991-10-22
	10029	1956-12-13	Otmar	Herbst	M	1985-11-20
	10030	1958-07-14	Elvis	Demeyer	M	1994-02-17
	10031	1959-01-27	Karsten	Joslin	M	1991-09-01
	10032	1960-08-09	Jeong	Reistad	F	1990-06-20
	10033	1956-11-14	Arif	Merlo	M	1987-03-18
	10034	1962-12-29	Bader	Swan	M	1988-09-21
	10035	1953-02-08	Alain	Chappelet	M	1988-09-05
	10036	1959-08-10	Adamantios	Portugali	M	1992-01-03
	10037	1963-07-22	Pradeep	Makrucki	M	1990-12-05
	10038	1960-07-20	Huan	Lortz	M	1989-09-20
	10039	1959-10-01	Alejandro	Brender	M	1988-01-19
	10040	1959-09-13	Weiyi	Meriste	F	1993-02-14
	10041	1959-08-27	Uri	Lenart	F	1989-11-12
	10042	1956-02-26	Magy	Stamatiou	F	1993-03-21
	10043	1960-09-19	Yishay	Tzvieli	M	1990-10-20
	10044	1961-09-21	Mingsen	Casley	F	1994-05-21
	10045	1957-08-14	Moss	Shanbho...	M	1989-09-02
	10046	1960-07-23	Lucien	Rosenbaum	M	1992-06-20
	10047	1952-06-29	Zvonko	Nyanchama	M	1989-03-31
	10048	1963-07-11	Florian	Syrotiuk	M	1985-02-24
	10049	1961-04-24	Basil	Tramer	F	1992-05-04
	10050	1958-05-21	Yinghua	Dredoe	M	1990-12-25

Soal 2 dan Query:

```
-- 2. Tampilkan judul buku yang termasuk dalam kategori komputer.  
-- Perubahan Soal : Tampilkan nama petugas dengan gender laki-laki
```

```
SELECT emp_no, first_name, last_name  
FROM employee  
WHERE gender = 'M';
```

Hasil run query:

Result Grid		Filter Rows:
emp_no	first_name	last_name
10001	Georgi	Facello
10003	Parto	Bamford
10004	Christian	Koblick
10005	Kyoichi	Maliniak
10008	Saniya	Kalloufi
10012	Patricio	Bridgland
10013	Eberhardt	Terkki
10014	Berni	Genin
10015	Guoxiang	Nooteboom
10016	Kazuhito	Cappelletti
10019	Lillian	Haddadi
10020	Mayuko	Warwick
10021	Ramzi	Erde
10022	Shahaf	Famili
10025	Prasadram	Heyers
10026	Yongqiao	Bertziss
10028	Domenick	Tempesti

Result Grid		Filter Rows:
emp_no	first_name	last_name
10029	Otmar	Herbst
10030	Elvis	Demeyer
10031	Karsten	Joslin
10033	Arif	Merlo
10034	Bader	Swan
10035	Alain	Chappelet
10036	Adamantios	Portugali
10037	Pradeep	Makrucki
10038	Huan	Lortz
10039	Alejandro	Brender
10043	Yishay	Tzvieli
10045	Moss	Shanbhogue
10046	Lucien	Rosenbaum
10047	Zvonko	Nyanchama
10048	Florian	Syrotiuk
10050	Yinghua	Dredge
10051	Hidefumi	Caine

Soal 3 dan Query:

```
-- 3. Tampilkan nama anggota yg meminjam lebih dari 2 buku
-- Perubahan Soal : Tampilkan nama petugas yang lahir mulai tahun 1960
```

```
SELECT emp_no, first_name, last_name, gender, hire_date
FROM employee
WHERE birth_date >= '1960-01-01';
```

Hasil run query:

	emp_no	first_name	last_name	gender	hire_date
▶	10002	Bezalel	Simmel	F	1985-11-21
	10010	Duangkaew	Piveteau	F	1989-08-24
	10012	Patricio	Bridgland	M	1992-12-18
	10013	Eberhardt	Terkki	M	1985-10-20
	10016	Kazuhiro	Cappelletti	M	1995-01-27
	10021	Ramzi	Erde	M	1988-02-10
	10027	Divier	Reistad	F	1989-07-07
	10028	Domenick	Tempesti	M	1991-10-22
	10032	Jeong	Reistad	F	1990-06-20
	10034	Bader	Swan	M	1988-09-21
	10037	Pradeep	Makrucki	M	1990-12-05
	10038	Huan	Lortz	M	1989-09-20
	10043	Yishay	Tzvieli	M	1990-10-20
	10044	Mingsen	Casley	F	1994-05-21
	10046	Lucien	Rosenbaum	M	1992-06-20
	10048	Florian	Syrotiuk	M	1985-02-24
	10049	Basil	Tramer	F	1992-05-04
	10052	Heping	Nitsch	M	1988-05-21
	10056	Brendon	Bernini	F	1990-02-01
	10060	Breannda	Billingsley	M	1987-11-02

	emp_no	first_name	last_name	gender	hire_date
	10060	Breannda	Billingsley	M	1987-11-02
	10061	Tse	Herber	M	1985-09-17
	10062	Anoosh	Peyn	M	1991-08-30
	10065	Satosi	Awdeh	M	1988-05-18
	10068	Charlene	Brattka	M	1987-08-07
	10069	Margareta	Bierman	F	1989-11-05
	10075	Gao	Dolinsky	F	1987-03-19
	10077	Mona	Azuma	M	1990-03-02
	10079	Kshitij	Gils	F	1986-03-27
	10081	Zhongwei	Rosen	M	1986-10-30
	10082	Parviz	Lortz	M	1990-01-03
	10084	Tuval	Kalloufi	M	1995-12-15
	10085	Kenroku	Malabarba	M	1994-04-09
	10086	Somnath	Foote	M	1990-02-16
	10089	Sudharsan	Flasterstein	F	1986-08-12
	10090	Kendra	Hofing	M	1986-03-14
	10092	Valdiodio	Niizuma	F	1989-09-22
	10093	Sailaja	Desikan	M	1996-11-05
	10095	Hilari	Morton	M	1986-07-15
	10098	Sreekrishna	Servieres	F	1985-05-13
*	NULL	NULL	NULL	NULL	NULL

Soal 4 dan Query:

```
-- 4. tampilkan nama anggota yang terlambat mengembalikan buku  
-- Perubahan Soal : Tampilkan nama pegawai laki-laki yang lahir di tahun 1960 yang di hire ditahun 1995.
```

```
SELECT emp_no, first_name, last_name, birth_date, hire_date  
FROM employee  
WHERE YEAR(birth_date) = 1960  
AND YEAR(hire_date) = 1995  
AND gender = 'M';
```

Hasil run query:

emp_no	first_name	last_name	birth_date	hire_date
10084	Tuval	Kalloufi	1960-05-25	1995-12-15
NULL	NULL	NULL	NULL	NULL

Soal 5 dan Query:

```
148 -- 5 Tampilkan nama anggota, nama jurusan yang mahasiswanya terlambat mengembalikan buku  
149 -- Perubahan Soal : Tampilkan nama dan jenis kelamin petugas yang mulai bekerja sebelum tahun 1986  
150  
151 • SELECT first_name, last_name, gender FROM employee WHERE hire_date < '1986-01-01';
```

Hasil run query:

first_name	last_name	gender
Bezalel	Simmel	F
Sumant	Peac	F
Eberhardt	Terkki	M
Otmar	Herbst	M
Florian	Syrotiuk	M
Tse	Herber	M
Udi	Jansch	M
Reuven	Garigliano	M
Erez	Ritzmann	F
Premal	Baek	M
Sreekrishna	Servieres	F

Soal 6 dan Query:

```
153 -- 6. Tampilkan nama anggota dan judul buku yang terlambat dikembalikan  
154 -- Perubahan Soal : Tampilkan nama petugas dan tanggal bekerja (hire date) yang di hire atau bekerja diatas tahun 1990  
155  
156 • SELECT emp_no, first_name, last_name, hire_date FROM employee WHERE hire_date >= '1990-01-01';
```


Hasil *run query*:

Result Grid				
Filter Rows:				
	emp_no	first_name	last_name	hire_date
▶	10008	Saniya	Kalloufi	1994-09-15
	10011	Mary	Sluis	1990-01-22
	10012	Patricio	Bridgland	1992-12-18
	10016	Kazuhito	Cappelletti	1995-01-27
	10017	Cristinel	Bouloucos	1993-08-03
	10019	Lillian	Haddadi	1999-04-30
	10020	Mayuko	Warwick	1991-01-26
	10022	Shahaf	Famili	1995-08-22
	10024	Suzette	Pettey	1997-05-19
	10026	Yongqiao	Bertziss	1995-03-20
	10028	Domenick	Tempesti	1991-10-22
	10030	Elvis	Demeyer	1994-02-17
	10031	Karsten	Joslin	1991-09-01
	10032	Jeong	Reistad	1990-06-20
	10036	Adamantios	Portugali	1992-01-03
	10037	Pradeep	Makrudi	1990-12-05
	10040	Weiyi	Meriste	1993-02-14
	10042	Magy	Stamatiou	1993-03-21
	10043	Yishay	Tzvieli	1990-10-20
	10044	Mingsen	Casley	1994-05-21
	10046	Lucien	Rosenbaum	1992-06-20
	10049	Basil	Tramer	1992-05-04

D. PEMBAHASAN

1. DML (Data Manipulation Language)

DML (Data Manipulation Language) adalah bahasa yang memungkinkan pengguna mengakses atau memanipulasi data pada database. Manipulasi data yang dimaksud :

1. Pengambilan informasi yang disimpan dalam database
2. Penempatan informasi baru dalam database
3. Penghapusan informasi dari database
4. Modifikasi informasi yang disimpan dalam database

2. Fungsi-fungsi yang ada dalam DML

1. SELECT

- Definisi : Fungsi untuk mengambil data dari suatu tabel pada database.
- Sintaks : `SELECT {* | daftar_kolom} FROM nama_tabel [WHERE kondisi_where]`
- Keterangan :
 1. Select digunakan untuk menampilkan semua kolom yang ada dalam tabel.
 2. Terdapat daftar_kolom yang dapat digunakan untuk mengambil kolom tertentu saja.
 3. WHERE digunakan jika ingin membatasi data yang ditampilkan.

2. INSERT

- Definisi: Fungsi untuk menambahkan data ke suatu tabel pada database.
- Sintaks : `INSERT INTO nama_tabel [(nama_kolom1[, nama_kolom2, ...])] VALUES (isi_kolom1[, isi_kolom2, ...])`
- Keterangan :
 1. Daftar nama kolom boleh ditulis, boleh tidak. Jika tidak ditulis, maka dianggap sesuai urutan nama kolom.
 2. Isi kolom harus sesuai dengan urutan daftar nama kolom. Gunakan koma sebagai pemisah.

3. UPDATE

- Definisi: Fungsi untuk mengubah data dari suatu tabel pada database.
- Sintaks : `UPDATE nama_tabel SET kolom1={isi|DEFAULT}[, kolom2={expr2|DEFAULT}] ... [WHERE kondisi_where]`
- Keterangan : WHERE digunakan untuk membatasi banyaknya baris yang diupdate.

4. DELETE

- Definisi: Fungsi untuk menghapus satu atau lebih baris data dari tabel di database.
- Sintaks : `DELETE FROM (nama_tabel) where`
- Keterangan : WHERE digunakan untuk membatasi banyaknya baris yang dihapus.

3. Penjelasan Query pada Soal

1. **Select* From Employee;** Query ini meminta untuk menampilkan semua kolom (*) dari tabel Employee.
2. **Select emp_no, first_name, last_name:** Ini menentukan kolom-kolom yang ingin ditampilkan. yakni kolom emp_no, first_name, dan last_name.
From employee: Ini menunjukkan bahwa data yang akan diambil berasal dari tabel yang bernama employee.
WHERE gender = 'M': Ini adalah klausa yang menyaring hasil. Hanya baris yang memiliki nilai gender sama dengan 'M' yang akan ditampilkan. Dalam konteks ini, 'M' merujuk pada karyawan pria.
3. **Select emp_no, first_name, last_name, gender, hire_date:** Ini menentukan kolom-kolom yang ingin ditampilkan. yakni kolom emp_no, first_name, last_name, gender, dan hire_date.
From employee: Ini menunjukkan bahwa data yang akan diambil berasal dari tabel yang bernama employee.
WHERE birth_date>='1960-01-01: Ini menyatakan bahwa hanya baris dengan nilai birth_date (tanggal lahir) yang lebih besar dari atau sama dengan 1 Januari 1960 yang akan disertakan dalam hasil.

4. **Select emp_no, first_name, last_name, birth_date, hire_date:** Ini menentukan kolom-kolom yang ingin ditampilkan. yakni kolom emp_no, first_name, last_name, birth_date, dan hire_date.
From employee: Ini menunjukkan bahwa data yang akan diambil berasal dari tabel yang bernama employee.
WHERE YEAR(birth_date)= 1960: Menyaring data untuk hanya menampilkan karyawan yang lahir pada tahun 1960.
WHERE YEAR(hire_date)= 1995: Menambahkan syarat tambahan untuk hanya menampilkan karyawan yang hire_date (dipekerjakan) pada tahun 1995.
AND gender = 'M': Menyaring lebih lanjut untuk hanya menyertakan karyawan yang berjenis kelamin laki-laki.
5. **Select emp_no, first_name, last_name, gender :** Ini menentukan kolom-kolom yang ingin ditampilkan. yakni kolom emp_no, first_name, last_name dan gender.
From employee: Ini menunjukkan bahwa data yang akan diambil berasal dari tabel yang bernama employee.
WHERE birth_date <'1986-01-01': Ini menyatakan bahwa hanya baris yang memiliki nilai birth_date(tanggal lahir) yang kurang dari 1 Januari 1986 yang akan disertakan dalam hasil.
6. **Select emp_no, first_name, last_name, birth_date, hire_date:** Ini menentukan kolom-kolom yang ingin ditampilkan. yakni kolom emp_no, first_name, last_name, birth_date, dan hire_date.
From employee: Ini menunjukkan bahwa data yang akan diambil berasal dari tabel yang bernama employee.
WHERE hire_date>='1990-01-01: Ini menyatakan bahwa hanya baris dengan nilai hire_date (dipekerjakan) yang lebih besar dari atau sama dengan 1 Januari 1990 yang akan disertakan dalam hasil.

E. KESIMPULAN

DML (Data Manipulation Language) adalah sekumpulan perintah dalam SQL yang digunakan untuk mengakses dan memanipulasi data di dalam database. DML memungkinkan pengguna untuk mengambil informasi dari database (select), menambahkan informasi baru ke database (insert), mengubah informasi yang sudah ada (update), menghapus informasi dari database (delete). Fungsi-fungsi DML tersebut memiliki sintaks yang jelas dan memungkinkan pengguna untuk melakukan berbagai operasi pada data. Penggunaan klausa where membantu membatasi pengambilan, pembaruan dan penghapusan data.

Contoh query yang diberikan menunjukkan berbagai cara untuk mengambil dan menyaring data dari tabel employee, termasuk pemfilteran berdasarkan gender, tanggal lahir, dan tanggal perekrutan. Kesalahan sintaks, seperti penggunaan tanda baca yang tidak tepat, perlu diperhatikan untuk memastikan query dapat dijalankan dengan benar.

Secara keseluruhan, DML adalah alat yang sangat penting untuk interaksi dengan data dalam database, memungkinkan pengguna untuk melakukan operasi dasar yang diperlukan dalam pengelolaan informasi.

MODUL 3

FUNGSI AGREGAT

A. SOAL TUGAS PRAKTIKUM

1. Buatlah Query yang menerapkan fungsi : Sum , AVG , Min, Max, Count, Group by , Order By , Having, dan Concat (Masing – masing fungsi 1 contoh)
2. Format Penulisan per contoh :
 - a. Soal
 - b. Syntax Query
 - c. Screenshoot Tabel sebelum dan sesudah query di running

B. ANALISIS MASALAH

1. Bagaimana cara menggunakan Syntax Query SUM, AVG, MIN, MAX, COUNT, GROUB BY, HAVING, dan CONCAT?
2. Apakah penulisan syntax Query SUM, AVG, MIN, MAX, COUNT, GROUP BY, HAVING, dan CONCAT telah dilakukan dengan benar selama percobaan?

C. HASIL PRAKTIKUM

1. SUM

SUM adalah sebuah fungsi yang digunakan untuk menghitung jumlah (total atau sum) nilai di suatu kolom.

Soal

Tampilkan total salary!

```
-- Tampilkan total salary
```



Syntax

```
SELECT SUM(salary) AS total_gaji FROM salary;
```

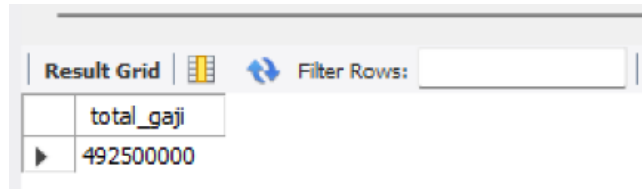
```
select sum(salary) as total_gaji from salary;
```

Jawab

Before

Result Grid   Filter Rows: <input type="text"/>				
	emp_no	salary	from_date	to_date
	10001	4500000	2023-01-01	2023-01-30
	10002	4800000	2023-01-01	2023-01-30
	10003	5200000	2023-01-01	2023-01-30
	10004	4700000	2023-01-01	2023-01-30
	10005	5000000	2023-01-01	2023-01-30
	10006	4600000	2023-01-01	2023-01-30
	10007	4900000	2023-01-01	2023-01-30
	10008	5100000	2023-01-01	2023-01-30
	10009	5300000	2023-01-01	2023-01-30
	10010	4400000	2023-01-01	2023-01-30
	10011	5200000	2023-01-01	2023-01-30

After



The screenshot shows a 'Result Grid' window with a 'Filter Rows' input field. Below the header, there is a single row with the column name 'total_gaji' and the value '492500000'.

	total_gaji
	492500000

2. AVG

AVG (Average) adalah sebuah fungsi yang digunakan untuk mengembalikan rata rata (average) nilai dari suatu kolom.

Soal

Tampilkan rata rata salary!

-- Tampilkan rata rata salary

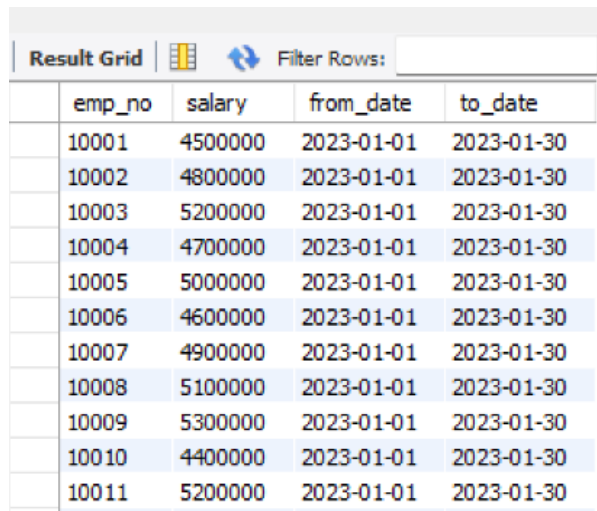
Syntax

SELECT AVG(salary) AS AverageSalary **FROM** salary;

`select avg(salary) as AverageSalary from salary;`

Jawab

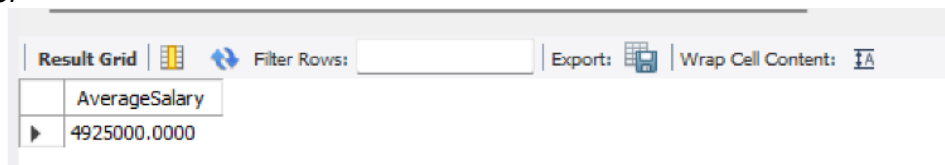
Before



The screenshot shows a 'Result Grid' window with a 'Filter Rows' input field. Below the header, there is a table with 5 columns: emp_no, salary, from_date, and to_date. The table contains 11 rows of employee data.

	emp_no	salary	from_date	to_date
	10001	4500000	2023-01-01	2023-01-30
	10002	4800000	2023-01-01	2023-01-30
	10003	5200000	2023-01-01	2023-01-30
	10004	4700000	2023-01-01	2023-01-30
	10005	5000000	2023-01-01	2023-01-30
	10006	4600000	2023-01-01	2023-01-30
	10007	4900000	2023-01-01	2023-01-30
	10008	5100000	2023-01-01	2023-01-30
	10009	5300000	2023-01-01	2023-01-30
	10010	4400000	2023-01-01	2023-01-30
	10011	5200000	2023-01-01	2023-01-30

After



The screenshot shows a 'Result Grid' window with a 'Filter Rows' input field. Below the header, there is a single row with the column name 'AverageSalary' and the value '4925000.0000'. The window also shows 'Export' and 'Wrap Cell Content' buttons.

	AverageSalary
	4925000.0000

3. MIN MAX

MIN adalah sebuah fungsi yang digunakan untuk mengembalikan nilai terkecil (minimal) suatu kolom.

MAX adalah sebuah fungsi yang digunakan untuk mengembalikan nilai terbesar (maksimal) suatu kolom.

Soal

Tampilkan data (hire_date) pegawai yang direkrut paling awal dan paling akhir berdasarkan gender!

-- Tampilkan data (hire_date) pegawai yang direkrut paling awal dan paling akhir berdasarkan gender

Syntax

SELECT (gender), **MAX**(hire_date) AS LatestHire, **MIN**(hire_date) AS EarliestHire **FROM** employee **GROUP BY** (gender)

`select (gender), max(hire_date) as LatestHire, min(hire_date) as EarliestHire from employee group by (gender);`

Jawab

Before

emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28
10004	1954-05-01	Christian	Koblick	M	1986-12-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
10006	1953-04-20	Anneke	Preusig	F	1989-06-02
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
10009	1952-04-19	Sumant	Peac	F	1985-02-18
10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24
10011	1953-11-07	Mary	Sluis	F	1990-01-22

After

gender	LatestHire	EarliestHire
M	1999-04-30	1985-02-24
F	1997-05-19	1985-02-18

4. COUNT

COUNT adalah sebuah fungsi yang digunakan untuk menghitung jumlah baris pada suatu kolom.

Soal

Tampilkan berapa jumlah karyawan yang memiliki gaji sebesar 5.000.000 atau lebih

-- Tampilkan gaji setiap karyawan dan hitung jumlah karyawan yang menerima gaji yang sama

Syntax

SELECT count(salary) AS count_high_salaries **FROM** salary **WHERE** salary >= 5000000;

`SELECT count(salary) AS count_high_salaries FROM salary WHERE salary >= 5000000;`

Jawab
Before

Result Grid				
Filter Rows:				
	emp_no	salary	from_date	to_date
	10001	4500000	2023-01-01	2023-01-30
	10002	4800000	2023-01-01	2023-01-30
	10003	5200000	2023-01-01	2023-01-30
	10004	4700000	2023-01-01	2023-01-30
	10005	5000000	2023-01-01	2023-01-30
	10006	4600000	2023-01-01	2023-01-30
	10007	4900000	2023-01-01	2023-01-30
	10008	5100000	2023-01-01	2023-01-30
	10009	5300000	2023-01-01	2023-01-30
	10010	4400000	2023-01-01	2023-01-30
	10011	5200000	2023-01-01	2023-01-30

After

Result Grid	
Filter Rows:	
	count_high_salaries
▶	47

5. GROUP BY

GROUP BY adalah sebuah fungsi yang digunakan untuk menggabungkan data dengan nilai yang sama dalam daftar bidang tertentu menjadi satu data.

Soal

Tampilkan gaji setiap karyawan dan hitung jumlah karyawan yang menerima gaji yang sama

-- Tampilkan gaji setiap karyawan dan hitung jumlah karyawan yang menerima gaji yang sama

Syntax

SELECT salary, **count**(*) AS salary_frequency **FROM** salary **GROUP BY** salary;

SELECT salary, **count**(*) AS salary_frequency **FROM** salary **GROUP BY** salary;

Jawab
Before

emp_no	salary	from_date	to_date
10001	4500000	2023-01-01	2023-01-30
10002	4800000	2023-01-01	2023-01-30
10003	5200000	2023-01-01	2023-01-30
10004	4700000	2023-01-01	2023-01-30
10005	5000000	2023-01-01	2023-01-30
10006	4600000	2023-01-01	2023-01-30
10007	4900000	2023-01-01	2023-01-30
10008	5100000	2023-01-01	2023-01-30
10009	5300000	2023-01-01	2023-01-30
10010	4400000	2023-01-01	2023-01-30
10011	5200000	2023-01-01	2023-01-30

After

salary	salary_frequency
4500000	9
4800000	9
5200000	11
4700000	11
5000000	10
4600000	12
4900000	11
5100000	10
5300000	12
4400000	1
5500000	2
5400000	2

6. ORDER BY

ORDER BY adalah sebuah fungsi yang digunakan untuk mengurutkan data dalam daftar bidang tertentu.

Soal

Tampilkan nomor dan gaji pegawai, diurutkan dari gaji yang terbesar hingga terkecil



```
-- Tampilkan nomor dan gaji pegawai, diurutkan dari gaji yang terbesar hingga terkecil
```

Syntax





```
SELECT emp_no, salary  
FROM salary  
ORDER BY salary DESC;
```

```
SELECT emp_no, salary FROM salary ORDER BY salary DESC;
```


Jawab
Before

Result Grid   Filter Rows: <input type="text"/>				
	emp_no	salary	from_date	to_date
	10001	4500000	2023-01-01	2023-01-30
	10002	4800000	2023-01-01	2023-01-30
	10003	5200000	2023-01-01	2023-01-30
	10004	4700000	2023-01-01	2023-01-30
	10005	5000000	2023-01-01	2023-01-30
	10006	4600000	2023-01-01	2023-01-30
	10007	4900000	2023-01-01	2023-01-30
	10008	5100000	2023-01-01	2023-01-30
	10009	5300000	2023-01-01	2023-01-30
	10010	4400000	2023-01-01	2023-01-30
	10011	5200000	2023-01-01	2023-01-30

After

Result Grid   Filter Rows: <input type="text"/>			Result Grid   Filter Rows: <input type="text"/>		
	emp_no	salary		emp_no	salary
▶	10081	5500000		10090	5300000
	10088	5500000		10067	5200000
	10082	5400000		10078	5200000
	10095	5400000		10003	5200000
	10017	5300000		10036	5200000
	10043	5300000		10028	5200000
	10034	5300000		10086	5200000
	10080	5300000		10094	5200000
	10023	5300000		10048	5200000
	10009	5300000		10022	5200000
	10058	5300000		10011	5200000
	10069	5300000		10100	5200000
	10051	5300000		10025	5100000
	10063	5300000		10032	5100000
	10097	5300000		10008	5100000
	10090	5300000		10084	5100000

7. HAVING

HAVING adalah sebuah fungsi yang digunakan untuk menggantikan klausa WHERE, karena klausa WHERE tidak boleh mengandung fungsi agregat.

Soal

Tampilkan total gaji karyawan yang dibayarkan pada bulan (to_date) januari atau bulan = 01.

-- Tampilkan total gaji karyawan yang dibayarkan pada bulan (to_date) Januari atau bulan = 01.

Syntax

```
SELECT MONTH(to_date) as bulan, SUM(salary) AS total_gaji  
FROM salary  
GROUP BY MONTH(to_date)  
HAVING bulan = 1 ;
```

```
SELECT MONTH(to_date) as bulan, SUM(salary) AS total_gaji  
FROM salary  
GROUP BY MONTH(to_date)  
HAVING bulan = 1;
```

Jawab

Before

Result Grid					Filter Rows:	
	emp_no	salary	from_date	to_date		
	10001	4500000	2023-01-01	2023-01-30		
	10002	4800000	2023-01-01	2023-01-30		
	10003	5200000	2023-01-01	2023-01-30		
	10004	4700000	2023-01-01	2023-01-30		
	10005	5000000	2023-01-01	2023-01-30		
	10006	4600000	2023-01-01	2023-01-30		
	10007	4900000	2023-01-01	2023-01-30		
	10008	5100000	2023-01-01	2023-01-30		
	10009	5300000	2023-01-01	2023-01-30		
	10010	4400000	2023-01-01	2023-01-30		
	10011	5200000	2023-01-01	2023-01-30		

After

Result Grid			Filter Rows:	
	bulan	total_gaji		
▶	1	492500000		

8. CONCAT

CONCAT adalah fungsi yang digunakan untuk menyambung string atau menggabungkan string hasil query.

Soal

Tampilkan nama lengkap pegawai dengan menggabungkan kolom firstname dan lastname.

```
Tampilkan nama lengkap pegawai dengan menggabungkan  
kolom firstname dan lastname
```

Syntax

SELECT CONCAT (first_name, ' ', last_name) AS nama_lengkap **FROM** employee;

```
SELECT CONCAT (first_name, ' ', last_name) AS nama_lengkap
FROM employee;
```

Jawab

Before

emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28
10004	1954-05-01	Christian	Koblick	M	1986-12-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
10006	1953-04-20	Anneke	Preusig	F	1989-06-02
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
10009	1952-04-19	Sumant	Peac	F	1985-02-18
10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24
10011	1953-11-07	Mary	Sluis	F	1990-01-22

After

nama_lengkap
Georgi Facello
Bezalel Simmel
Parto Bamford
Christian Koblick
Kyoichi Maliniak
Anneke Preusig
Tzvetan Zielinski
Saniya Kalloufi
Sumant Peac
Duangkaew Piveteau

D. PEMBAHASAN

1. Definisi Fungsi Agregat

Fungsi agregat adalah fungsi dalam SQL yang digunakan untuk melakukan operasi perhitungan pada sekelompok data dan mengembalikan nilai tunggal sebagai hasilnya. Fungsi ini sangat berguna untuk menganalisis data dalam tabel, seperti menghitung total, rata-rata, jumlah data, nilai maksimum, dan minimum. Fungsi agregat biasanya

digunakan bersama dengan perintah GROUP BY untuk mengelompokkan data berdasarkan kategori tertentu.

2. Jenis Fungsi Agregat

a. SUM

Definisi :Menghitung jumlah total nilai dari setiap baris pada suatu kolom

Query : **SELECT SUM(nama_field)FROM** TABLE;

b. AVG

Definisi : Menghitung rata-rata (average) nilai pada suatu kolom

Query : **SELECT AVG(nama_field)FROM** TABLE;

c. MIN

Definisi : Meghitung nilai terkecil (minimal) pada suatu kolom

Query : **SELECT MIN(nama_field)FROM** TABLE;

d. MAX

Definisi : Menghitung nilai terbesar (maximal) pada suatu kolom

Query : **SELECT MAX(nama_field)FROM** TABLE;

e. COUNT

Definisi : Menghitung jumlah baris pada suatu kolom

Query : **SELECT COUNT(nama_field)FROM** TABLE;

f. GROUP BY

Definisi : GROUP BY digunakan untuk menggabungkan data dengan nilai yang sama dalam daftar bidang tertentu menjadi satu data.

Query : **SELECT(namaField)FROM** table **GROUP BY** kategori;

g. ORDER BY

Definisi : ORDER BY adalah sebuah fungsi yang digunakan untuk mengurutkan data dalam daftar bidang tertentu.

Query : **SELECT** kolom1, kolom2 **FROM** nama_tabel **ORDER BY** kolom1 [ASC|DESC];

h. HAVING

Definisi : HAVING adalah klausa yang digunakan untuk memfilter hasil agregasi setelah data dikelompokkan.

Query : **SELECT** field1,field..n, **.Aggregate_function FROM** TABLE **GROUP BY** field1,field..n **HAVING** kriteria_aggerage_function;

i. CONCAT

Definisi : Fungsi ini digunakan untuk menyambung string atau menggabungkan string hasil query.

Query : **SELECT CONCAT** field1,field..n **FROM** TABLE;

3. Pembahasan Query pada Praktikum

a. SUM

soal : Tampilkan total salary!

Query : **SELECT SUM(salary) AS total_gaji FROM** salary;

Pembahasan Query :

1. **SELECT**: Ini digunakan untuk memilih data dari tabel di database.
2. **SUM(salary)**: Fungsi agregasi SUM() menjumlahkan semua nilai dari kolom salary. Dalam contoh ini, semua nilai dalam kolom salary akan dijumlahkan.
3. **AS total_gaji**: Menggunakan alias AS untuk memberi nama baru pada kolom hasil, sehingga hasil penjumlahan ditampilkan dengan nama total_gaji.

4. **FROM** salary: Menentukan tabel asal, yaitu tabel salary, tempat data akan diambil.

b. AVG

soal : Tampilkan rata rata salary!

Query : **SELECT AVG**(salary) AS AverageSalary **FROM** salary;

Pembahasan Query :

1. **SELECT**: Digunakan untuk memilih data dari tabel dalam database.
2. **AVG**(salary): Fungsi agregasi AVG() menghitung **rata-rata** dari semua nilai dalam kolom salary. Rata-rata dihitung dengan menjumlahkan seluruh nilai di kolom salary lalu membaginya dengan jumlah baris (data) dalam kolom tersebut.
3. AS AverageSalary: Menggunakan **alias** (AS) untuk memberikan nama pada kolom hasil agar lebih mudah dibaca. Dalam contoh ini, hasil rata-rata ditampilkan dengan nama AverageSalary.
4. **FROM** salary: Menentukan bahwa data diambil dari tabel salary.

c. MIN dan MAX

soal : Tampilkan data (hire_date) pegawai yang direkrut paling awal dan paling akhir berdasarkan gender!

Query : **SELECT** (gender), **MAX**(hire_date) AS LatestHire, **MIN**(hire_date) AS EarliestHire **FROM** employee **GROUP BY** (gender)

Pembahasan Query :

1. **SELECT**: Digunakan untuk memilih data dari tabel.
2. (gender): Menyatakan bahwa kita ingin mengambil kolom gender sebagai salah satu kolom yang akan ditampilkan dalam hasil query.
3. **MAX**(hire_date) AS LatestHire: Fungsi agregasi MAX() digunakan untuk mendapatkan tanggal perekrutan (hire date) paling akhir. Dengan menggunakan alias AS LatestHire, hasilnya akan ditampilkan dengan nama yang lebih deskriptif.
4. **MIN**(hire_date) AS EarliestHire: Fungsi agregasi MIN() digunakan untuk mendapatkan tanggal perekrutan paling awal. Alias AS EarliestHire juga digunakan untuk memberikan nama yang jelas pada hasil.
5. **FROM** employee: Menunjukkan bahwa data yang diambil berasal dari tabel employee, yang berisi informasi tentang pegawai.
6. **GROUP BY** (gender): Digunakan untuk mengelompokkan hasil berdasarkan kolom gender. Ini berarti bahwa hasil query akan menampilkan satu baris untuk setiap jenis kelamin, dengan LatestHire dan EarliestHire untuk masing-masing kelompok gender.

d. COUNT

soal : Tampilkan berapa jumlah karyawan yang memiliki gaji sebesar 5.000.000 atau lebih

Query : **SELECT count**(salary) AS count_high_salaries **FROM** salary **WHERE** salary >= 5000000;

Pembahasan Query :

1. **SELECT**: Digunakan untuk memilih data dari tabel di database.
2. **count(salary)**: Fungsi agregasi **count()** digunakan untuk menghitung jumlah baris (entri) yang memenuhi kriteria tertentu. Dalam hal ini, fungsi ini menghitung jumlah pegawai yang memiliki gaji di atas atau sama dengan 5.000.000.
3. **AS count_high_salaries**: Menggunakan alias **AS** untuk memberikan nama pada kolom hasil, sehingga hasil dari perhitungan ditampilkan dengan nama **count_high_salaries**. Ini memudahkan pemahaman tentang apa yang dihitung.
4. **FROM salary**: Menunjukkan bahwa data yang diambil berasal dari tabel **salary**, yang berisi informasi tentang gaji karyawan.
5. **WHERE salary >= 5000000**: Kondisi ini memfilter data sehingga hanya gaji yang lebih besar dari atau sama dengan 5.000.000 yang dihitung. Ini memastikan bahwa hanya karyawan dengan gaji tersebut yang termasuk dalam perhitungan.

e. **GROUP BY**

soal : Tampilkan gaji setiap karyawan dan hitung jumlah karyawan yang menerima gaji yang sama

Query : **SELECT salary, count(*) AS salary_frequency FROM salary GROUP BY salary;**

Pembahasan Query :

1. **SELECT**: Digunakan untuk memilih data dari tabel dalam database.
2. **salary**: Menyatakan bahwa kita ingin menampilkan kolom **salary**, yang berisi informasi tentang gaji setiap karyawan.
3. **count(*) AS salary_frequency**: Fungsi agregasi **count(*)** menghitung jumlah baris (karyawan) untuk setiap nilai gaji yang sama. Menggunakan alias **AS salary_frequency**, hasilnya akan ditampilkan dengan nama yang menunjukkan frekuensi atau jumlah karyawan yang menerima gaji yang sama.
4. **FROM salary**: Menunjukkan bahwa data yang diambil berasal dari tabel **salary**, yang berisi informasi tentang gaji karyawan.
5. **GROUP BY salary**: Digunakan untuk mengelompokkan hasil berdasarkan nilai gaji. Ini berarti bahwa hasil query akan menampilkan satu baris untuk setiap nilai gaji unik, dengan jumlah karyawan yang menerima gaji tersebut.

f. **ORDER BY**

soal : Tampilkan nomor dan gaji pegawai, diurutkan dari gaji yang terbesar hingga terkecil

Query : **SELECT emp_no, salary FROM salary**

ORDER BY salary DESC;

Pembahasan Query :

1. **SELECT**: Digunakan untuk memilih data dari tabel dalam database.
2. **emp_no**: Menyatakan bahwa kita ingin menampilkan kolom **emp_no**, yang berisi nomor identifikasi pegawai.

3. **salary**: Menyatakan bahwa kita juga ingin menampilkan kolom salary, yang berisi informasi tentang gaji pegawai.
4. **FROM salary**: Menunjukkan bahwa data yang diambil berasal dari tabel salary, yang menyimpan informasi tentang pegawai dan gaji mereka.
5. **ORDER BY salary DESC**: Mengurutkan hasil query berdasarkan kolom salary. Kata kunci DESC berarti bahwa hasil akan diurutkan dari gaji yang terbesar hingga yang terkecil. Jika ingin mengurutkan dari yang terkecil hingga yang terbesar, kita dapat menggunakan ASC (ascending) sebagai gantinya.

g. **HAVING**

soal : Tampilkan total gaji karyawan yang dibayarkan pada bulan (to_date) januari atau bulan = 01.

Query : **SELECT MONTH (to_date) as bulan, SUM(salary) AS total_gaji**

FROM salary

GROUP BY MONTH(to_date)

HAVING bulan = 1 ;

Pembahasan Query :

1. **SELECT**: Digunakan untuk memilih data dari tabel.
2. **MONTH(to_date) AS bulan**: Fungsi MONTH() digunakan untuk mengekstrak nomor bulan dari kolom to_date, yang menyimpan informasi tanggal pembayaran gaji. Alias AS bulan memberikan nama baru pada kolom hasil untuk memperjelas bahwa kolom tersebut menunjukkan bulan.
3. **SUM(salary) AS total_gaji**: Fungsi agregasi SUM() digunakan untuk menghitung total gaji yang dibayarkan pada bulan yang ditentukan. Alias AS total_gaji memberikan nama yang jelas pada hasil penjumlahan.
4. **FROM salary**: Menunjukkan bahwa data diambil dari tabel salary, yang menyimpan informasi tentang gaji karyawan.
5. **GROUP BY MONTH(to_date)**: Digunakan untuk mengelompokkan hasil berdasarkan bulan yang diekstrak dari kolom to_date. Ini memastikan bahwa total gaji dihitung untuk setiap bulan secara terpisah.
6. **HAVING bulan = 1**: Kondisi ini digunakan untuk memfilter hasil agregasi. Hanya hasil dengan bulan yang sama dengan 1 (Januari) yang akan ditampilkan. Perlu dicatat bahwa HAVING digunakan setelah GROUP BY untuk memfilter hasil agregasi, sedangkan WHERE digunakan sebelum pengelompokan.

h. **CONCAT**

soal : Tampilkan nama lengkap pegawai dengan menggabungkan kolom firstname dan lastname

Query : **SELECT CONCAT (first_name, ' ', last_name) AS nama_lengkap FROM employee;**

Pembahasan Query :

1. **SELECT**: Digunakan untuk memilih data dari tabel dalam database.

2. **CONCAT**(first_name, ' ', last_name): Fungsi **CONCAT**() digunakan untuk menggabungkan dua atau lebih string. Dalam query ini, kolom first_name dan last_name digabungkan dengan menambahkan spasi di antara keduanya (' '). Hasil dari fungsi ini adalah nama lengkap pegawai yang terdiri dari nama depan dan nama belakang.
3. **AS** nama_lengkap: Menggunakan alias **AS** untuk memberikan nama baru pada kolom hasil, sehingga hasil gabungan ditampilkan dengan nama nama_lengkap. Ini memudahkan pembacaan dan pemahaman hasil query.
4. **FROM** employee: Menunjukkan bahwa data yang diambil berasal dari tabel employee, yang menyimpan informasi tentang pegawai.

E. KESIMPULAN

Fungsi agregat seperti **SUM**, **AVG**, **MIN**, **MAX**, dan **COUNT** sangat membantu dalam menganalisis data numerik. Untuk menyajikan data dengan cara yang lebih informatif, pengelompokan (**GROUP BY**) dan filter setelah pengelompokan (**HAVING**) diperlukan. Selain itu, penggabungan string dengan **CONCAT** memungkinkan data teks lebih mudah dibaca. Dengan penerapan fungsi-fungsi tersebut, analisis dan pelaporan data menjadi lebih terstruktur dan efisien.

MODUL 4

ALJABAR RELASIONAL

A. SOAL TUGAS PRAKTIKUM

1. Pelajari Aljabar Relasional
2. Apa saja simbol simbol aljabar relasional?
3. Bagaimana penulisan aljabar relasional?
4. Jawablah pertanyaan berikut :
 - a. Tampilkan informasi pegawai yang ada di sebuah PT X.
 - b. Tampilkan nama petugas dengan gender laki laki.
 - c. Tampilkan nama petugas yang lahir mulai tahun 1960.
 - d. Tampilkan nama pegawai laki laki yang lahir di tahun 1960 yang di-hire di tahun 1995.
 - e. Tampilkan nama dan jenis kelamin petugas yang mulai bekerja sebelum tahun 1986.

B. ANALISIS MASALAH

Aljabar Relasional adalah landasan teoritis untuk operasi pada basis data relasional. Aljabar relasional dikembangkan oleh Edgar F. Codd pada tahun 1970. Aljabar relasional digunakan untuk memanipulasi dan mengekstrak data. Pemahaman mengenai operasi operasi aljabar relasional sangat penting diterapkan terutama dalam melakukan evaluasi query secara efisien. Aljabar relasional merupakan materi yang erat kaitannya dengan bagaimana query SQL dipetakan dari operasi aljabar relasional, yang menjadi fokus dalam materi ini. Analisis akan mencakup permasalahan mengenai penerapan operasi seleksi (*select*), proyeksi (*project*) dan *cartesian product*. Analisis masalah yang telah ditemukan mencakup :

- Kesulitan dalam Memahami Operasi Seleksi dan Proyeksi
Operasi seleksi dan proyeksi merupakan operasi dasar dalam aljabar relasional, namun banyak yang mengalami kesulitan dalam memahami fungsi dan penggabungan dua operasi ini. Operasi seleksi (σ) digunakan untuk memilih baris tertentu dari tabel berdasarkan kondisi tertentu, sementara operasi proyeksi (π) digunakan untuk memilih kolom atau atribut tertentu dari suatu tabel.
- Kesalahan dalam Penggunaan Operasi *Cartesian Product*.
Cartesian Product (\times) adalah operasi yang menggabungkan setiap baris dari tabel pertama dengan setiap baris dari tabel kedua, menghasilkan kombinasi semua baris yang mungkin. *Cartesian Product* akan menghasilkan data yang tidak relevan dan berlebihan sehingga hasil yang didapatkan terlalu besar dan sulit untuk diolah.
- Kompleksitas Query
Query aljabar relasional yang kompleks akan sulit dibaca dan dipahami, terutama untuk basis data yang besar dan rumit. Hal ini memiliki dampak dalam pemeliharaan dan debugging sistem basis data.

C. HASIL PRAKTIKUM

Hasil pemetaan dari aljabar relasional ke query akan tampak sebagai berikut :

1. Tampilkan informasi pegawai yang ada di sebuah PT X.

- Aljabar Relasional

π emp_no , birth_date, first_name, last_name, gender, hire_date
(employee)

- Query

-- Perubahan Soal : Tampilkan informasi pegawai yang ada di sebuah PT X

```
SELECT * FROM employee;
```

2. Tampilkan nama petugas dengan gender laki laki.

- Aljabar Relasional

π emp_no, first_name, last_name (σ gender = 'M'(employee))

- Query

-- Perubahan Soal : Tampilkan nama petugas dengan gender laki-laki

```
SELECT emp_no, first_name, last_name
FROM employee
WHERE gender = 'M';
```

3. Tampilkan nama petugas yang lahir mulai tahun 1960.

- Aljabar Relasional

π emp_no, first_name, last_name, gender, hire_date (σ birth_date \geq '1960-01-01'(employee))

- Query

-- Perubahan Soal : Tampilkan nama petugas yang lahir mulai tahun 1960

```
SELECT emp_no, first_name, last_name, gender, hire_date
FROM employee
WHERE birth_date >= '1960-01-01';
```

4. Tampilkan nama pegawai laki laki yang lahir di tahun 1960 yang di-hire di tahun 1995.

- Aljabar Relasional

π emp_no, first_name, last_name, gender, birth_date, hire_date (σ birth_date = '1960' AND hire_date = '1995' AND gender = 'M' (employee))

- Query

-- Perubahan Soal : Tampilkan nama pegawai laki-laki yang lahir di tahun 1960 yang di hire ditahun 1995.

```
SELECT emp_no, first_name, last_name, birth_date, hire_date
FROM employee
WHERE YEAR(birth_date) = 1960
AND YEAR(hire_date) = 1995
AND gender = 'M';
```

5. Tampilkan nama dan jenis kelamin petugas yang mulai bekerja sebelum tahun 1986.

- Aljabar Relasional

π first_name, last_name, gender (σ hire_date < '1986-01-01' (employee))

- Query

-- Perubahan Soal : Tampilkan nama dan jenis kelamin petugas yang mulai bekerja sebelum tahun 1986

```
SELECT first_name, last_name, gender FROM employee WHERE hire_date < '1986-01-01';
```

6. Tampilkan nama petugas dan tanggal bekerja (hire_date) yang di-hire atau bekerja diatas tahun 1990

- Aljabar Relasional

π emp_no, first_name, last_name, gender, hire_date (σ hire_date >= '1990-01-01' (employee))

- Query

-- Perubahan Soal : Tampilkan nama petugas dan tanggal bekerja (hire date) yang di hire atau bekerja diatas tahun 1990

```
SELECT emp_no, first_name, last_name, hire_date FROM employee WHERE hire_date >= '1990-01-01';
```

D. PEMBAHASAN

Aljabar Relasional yang diperkenalkan oleh Edgar F. Codd pada tahun 1970 adalah sebuah bahasa query formal yang menjadi landasan teoritis untuk model basis data relasional. Aljabar relasional digunakan sebagai bahasa prosedural yang menentukan langkah langkah yang diperlukan untuk mendapatkan hasil yang diinginkan dari basis data. Namun penggunaan aljabar relasional tidak dapat diterapkan ke dalam aplikasi SQL yang digunakan untuk mendeklarasikan sebuah *database*. Karena aljabar relasional cenderung mendalam ke teori himpunan dan juga logika predikat, sedangkan SQL lebih bersifat deklaratif.

Hubungan antara aljabar relasional dengan query yaitu aljabar relasional merupakan bahasa formal dari query, aljabar relasional juga merupakan bahasa query yang diterjemah dalam bentuk simbol simbol matematis dan sangat bermanfaat untuk melakukan rencana evaluasi query yang sifatnya prosedural, aljabar relasional adalah dasar dari SQL. Sedangkan, SQL merupakan bahasa komersial dari query dan bersifat *user friendly* bagi *programmer* karena bersifat rasional.

Aljabar relasional memiliki beberapa jenis operasi dan juga operator yang digunakan untuk memanipulasi data, diantaranya :

1. Operasi Dasar Aljabar Relasional

- Selection (σ)
- Projection (π)
- Cartesian-product (\times , disebut juga cross product)
- Union (U)
- Set-difference (-)

2. Operasi Tambahan Aljabar Relasional

- Set intersection (\cap)
- Natural join (\bowtie)
- Theta join (θ)
- Division (\div)

Namun pada pembahasan kali ini akan berfokus pada 3 operasi dasar aljabar relasional, yaitu selection, projection dan juga cartesian product.

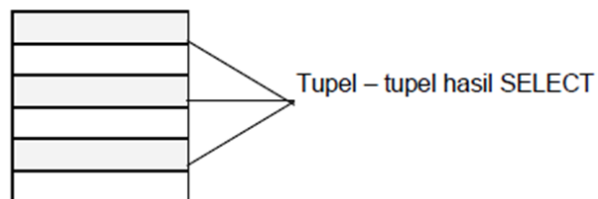
1. SELECTION

Operasi ini berfungsi untuk menyeleksi tuple yang memenuhi predikat tertentu dari suatu relasi. Operasi ini memiliki simbol σ (sigma) dan melibatkan :

- Operand : konstanta/bilangan
- Operator aritmatika : $<, =, >, \geq, \leq, \neq$
- Operator logika : \wedge (and), \vee (or), \neg (not)

Ilustrasi :

R



Operasi Selection memiliki sintaks sebagai berikut : $\sigma P(E)$ dengan P adalah predikat pada atribut E dan E adalah tabel atau relasi.

Contoh :

Tampilkan employee dengan age<24

"employee" table			
em_id	name	gender	age
121	Jerico	M	25
122	Maxime	M	25
123	Alexander	M	26
124	Gracia	F	22

Operasi : $\sigma \text{ age} < 24^{(\text{employee})}$

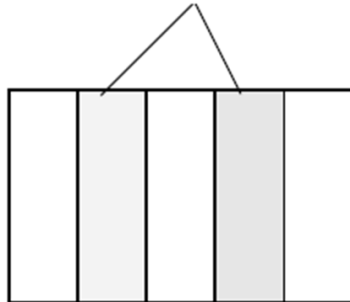
Hasil :

124	Gracia	F	22
-----	--------	---	----

2. PROJECTION

Proyeksi adalah operasi untuk memperoleh kolom-kolom tertentu untuk ditampilkan. Operasi proyeksi merupakan operasi unary yang mengirim relasi argumen dengan kolom-kolom tertentu. Operasi ini memiliki simbol Π (phi).

Ilustrasi : **project**



Operasi Selection memiliki sintaks sebagai berikut :
 π **column1, ... ,columnN** (tabel)

Contoh :

Tampilkan nama dan age dari tabel employee

"employee" table			
em_id	name	gender	age
121	Jerico	M	25
122	Maxime	M	25
123	Alexander	M	26
124	Gracia	F	22

Operasi : π nama, age (employee)

Hasil :

name	age
Jerico	25
Maxime	25
Alexander	26
Gracia	22

3. GABUNGAN SELECTION DAN PROJECTION

Seleksi dan proyeksi digunakan saat query yang digunakan berada dalam klausa WHERE, sehingga query tersebut menjadi query bersyarat. dalam klausa WHERE terdapat operator yang bisa digunakan, yaitu :

Operator	Description
=	Equal (sama dengan)
!=	Not equal (tidak sama dengan)
>	Greater than (Lebih dari)
<	Less than (kurang dari)
>=	Greater than or equal (lebih dari sama dengan)
<=	Less than or equal (kurang dari sama dengan)
BETWEEN	Between an inclusive range
LIKE	Search for a pattern (pencarian dengan pola tertentu)

Contoh Pemetaan aljabar relasional gabungan ke dalam query SQL :
Tampilkan nama dan gender pegawai yang umurnya >23

"employee" table			
em_id	name	gender	age
121	Jerico	M	25
122	Maxime	M	25
123	Alexander	M	26
124	Gracia	F	22

Aljabar Relasional

π nama, gender (σ age > 23_(employee))

Query SQL

SELECT nama, gender FROM employee WHERE age > 23

Hasil

name	gender
Jerico	M
Maxime	M
Alexander	M

4. CARTESIAN PRODUCT

Operasi Cartesian Product digunakan untuk merelasikan semua record record yang berasal dari dua tabel. Tujuan dari cartesian product adalah membentuk suatu relasi dari dua relasi yang terdiri dari kombinasi tuple tuple yang mungkin. Operasi ini memiliki simbol \times (cross). Operasi cartesian product umumnya tidak berdiri sendiri karena akan menghasilkan data data yang tidak valid dan tidak relevan, tetapi dapat digunakan bersama operasi lainnya seperti select dan project agar tidak terjadi kekacauan pada database.

Ilustrasi :

R	S	R X S
a	1	a 1
a	2	a 2
a	3	a 3
b	1	b 1
b	2	b 2
b	3	b 3

Operasi ini memiliki sintaks : $E1 \times E2$

Semua record E1 akan dipasangkan dengan semua record E2. Operasi ini bersifat komutatif, artinya $E1 \times E2$ akan sama dengan $E2 \times E1$

Contoh :

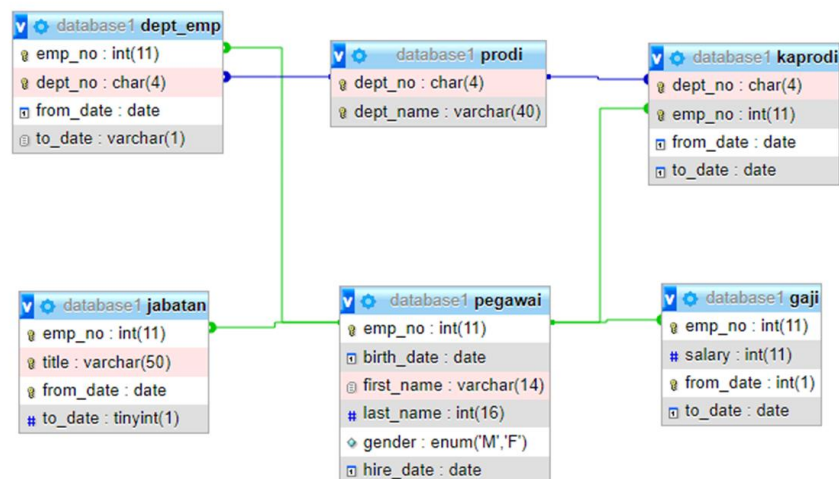
name	gender	salary
Jerico	M	3500
Maxime	M	1000
Gracia	F	

Hasil

name	gender	salary
Jerico	M	3500
Jerico	M	1000
Maxime	M	3500
Maxime	M	1000
Gracia	F	3500
Gracia	F	1000

- PEMBAHASAN HASIL PRAKTIKUM

Praktikum diambil dari database pegawai :



- Tampilkan informasi pegawai yang ada di sebuah PT X.

- Aljabar Relasional

π emp_no , birth_date, first_name, last_name, gender, hire_date
(employee)

- Query

-- Perubahan Soal : Tampilkan informasi pegawai yang ada di sebuah PT X

```
SELECT * FROM employee;
```

- Pembahasan :

Pada sintaks diatas diterapkan aljabar relasional projection (proyeksi) untuk menampilkan atribut atribut dalam tabel employee, sehingga saat dipetakan ke SQL menjadi **SELECT * FROM employee**, karena dari aljabar relasional nya meminta seluruh atribut di tabel employee untuk ditampilkan.

2. Tampilkan nama petugas dengan gender laki laki.

- Aljabar Relasional

π emp_no, first_name, last_name (σ gender = 'M'(employee))

- Query

-- Perubahan Soal : Tampilkan nama petugas dengan gender laki-laki

```
SELECT emp_no, first_name, last_name
FROM employee
WHERE gender = 'M';
```

- Pembahasan :

Pada sintaks diatas diterapkan aljabar relasional gabungan yaitu projection (proyeksi) dan selection (seleksi). Projection (π) digunakan untuk menampilkan atribut emp_no, first_name dan last_name dalam tabel employee, sedangkan selection (σ) digunakan untuk menyeleksi data yang memenuhi syarat untuk ditampilkan yaitu gender = 'M'. Sehingga saat dipetakan ke SQL menjadi **SELECT emp_no, first_name, last_name FROM employee WHERE gender = 'M'**, karena dari aljabar relasional nya ada seleksi data yang ditampilkan adalah data yang memiliki gender laki laki (male).

3. Tampilkan nama petugas yang lahir mulai tahun 1960.

- Aljabar Relasional

π emp_no, first_name, last_name, gender, hire_date (σ birth_date >= '1960-01-01'(employee))

- Query

-- Perubahan Soal : Tampilkan nama petugas yang lahir mulai tahun 1960

```
SELECT emp_no, first_name, last_name, gender, hire_date
FROM employee
WHERE birth_date>='1960-01-01';
```

- Pembahasan :

Pada sintaks diatas diterapkan aljabar relasional gabungan yaitu projection (proyeksi) dan selection (seleksi). Projection (π) digunakan untuk menampilkan atribut emp_no, first_name, last_name, gender, hire_date dalam tabel employee, sedangkan selection (σ) digunakan untuk menyeleksi data yang memenuhi syarat untuk ditampilkan yaitu birth_date>='1960-01-01'. Sehingga saat dipetakan ke SQL menjadi **SELECT emp_no, first_name, last_name, gender, hire_date FROM employee WHERE birth_date>='1960-01-01'**, karena dari aljabar relasional nya ada seleksi data yang ditampilkan adalah data yang memiliki tahun kelahiran diatas 1960.

4. Tampilkan nama pegawai laki laki yang lahir di tahun 1960 yang di-hire di tahun 1995.

- Aljabar Relasional

π emp_no, first_name, last_name, gender, birth_date, hire_date (σ birth_date = '1960' AND hire_date = '1995' AND gender = 'M' (employee))

- Query

-- Perubahan Soal : Tampilkan nama pegawai laki-laki yang lahir di tahun 1960 yang di hire ditahun 1995.

```
SELECT emp_no, first_name, last_name, birth_date, hire_date
FROM employee
WHERE YEAR(birth_date) = 1960
      AND YEAR(hire_date) = 1995
      AND gender = 'M';
```

- Pembahasan

Pada sintaks diatas diterapkan aljabar relasional gabungan yaitu projection (proyeksi) dan selection (seleksi). Projection (π) digunakan untuk menampilkan atribut emp_no, first_name, last_name, gender, birth_date, hire_date dalam tabel employee, sedangkan selection (σ) digunakan untuk menyeleksi data yang memenuhi syarat untuk ditampilkan yaitu birth_date = '1960' AND hire_date = '1995' AND gender = 'M'. Sehingga saat dipetakan ke SQL menjadi **SELECT emp_no, first_name, last_name, gender, birth_date, hire_date FROM employee WHERE birth_date = '1960' AND hire_date = '1995' AND gender = 'M'**, karena dari aljabar relasional nya ada seleksi data yang ditampilkan adalah data yang memiliki tahun kelahiran diatas 1960, di-hire di tahun 1995 dan memiliki gender laki laki (male).

5. Tampilkan nama dan jenis kelamin petugas yang mulai bekerja sebelum tahun 1986.

- Aljabar Relasional

π first_name, last_name, gender (σ hire_date < '1986-01-01' (employee))

- Query

-- Perubahan Soal : Tampilkan nama dan jenis kelamin petugas yang mulai bekerja sebelum tahun 1986

```
SELECT first_name, last_name, gender FROM employee WHERE hire_date < '1986-01-01';
```

- Pembahasan :

Pada sintaks diatas diterapkan aljabar relasional gabungan yaitu projection (proyeksi) dan selection (seleksi). Projection (π) digunakan untuk menampilkan atribut first_name, last_name dan gender dalam tabel employee, sedangkan selection (σ) digunakan untuk menyeleksi data yang memenuhi syarat untuk ditampilkan yaitu hire_date < '1986-01-01'. Sehingga saat dipetakan ke SQL menjadi **SELECT first_name, last_name, gender FROM employee WHERE hire_date < '1986-01-01'**, karena dari aljabar relasional nya ada seleksi data yang ditampilkan adalah data pegawai yang dihire sebelum tahun 1986.

6. Tampilkan nama petugas dan tanggal bekerja (hire_date) yang di-hire atau bekerja diatas tahun 1990

- Aljabar Relasional

π emp_no, first_name, last_name, gender, hire_date (σ hire_date >= '1990-01-01'(employee))

- Query

-- Perubahan Soal : Tampilkan nama petugas dan tanggal bekerja (hire date) yang di hire atau bekerja diatas tahun 1990

```
SELECT emp_no, first_name, last_name, hire_date FROM employee WHERE hire_date >= '1990-01-01';
```

- Pembahasan :

Pada sintaks diatas diterapkan aljabar relasional gabungan yaitu projection (proyeksi) dan selection (seleksi). Projection (π) digunakan untuk menampilkan atribut emp_no, first_name, last_name, gender dan hire_date dalam tabel employee, sedangkan selection (σ) digunakan untuk menyeleksi data yang memenuhi syarat untuk ditampilkan yaitu hire_date \geq '1990-01-01'. Sehingga saat dipetakan ke SQL menjadi **SELECT first_name, last_name, gender, hire_date FROM employee WHERE hire_date \geq '1990-01-01'**, karena dari aljabar relasional nya ada seleksi data yang ditampilkan adalah data pegawai yang dihire setelah tahun 1990.

E. KESIMPULAN

Pada praktikum ini, aljabar relasional terbukti menjadi landasan penting dalam manipulasi data pada basis data relasional. Melalui penerapan operasi-operasi aljabar relasional seperti seleksi (*select*), proyeksi (*project*), *cartesian product*, dan gabungan (*join*), dapat mengelola dan mengekstraksi data secara sistematis serta efisien. Aljabar relasional memungkinkan perancangan query yang kemudian diimplementasikan ke dalam bahasa SQL, yang merupakan bahasa query komersial yang digunakan dalam berbagai sistem manajemen basis data relasional.

Dalam praktikum ini, dipelajari cara memetakan operasi dasar aljabar relasional ke dalam sintaks SQL, termasuk seleksi data berdasarkan kondisi tertentu, pemilihan atribut spesifik, serta penggabungan data dari berbagai tabel dengan menggunakan operasi produk kartesian maupun gabungan. Pemahaman yang mendalam terhadap urutan operasi dalam menyusun kueri menjadi sangat penting untuk mendapatkan hasil yang sesuai dan optimal.

Selain itu, aspek optimasi query menjadi sorotan utama, mengingat efisiensi query merupakan faktor krusial dalam pengelolaan basis data berskala besar. Dengan simulasi menggunakan data uji, pengaruh signifikan dari pemilihan operasi yang tepat terhadap performa kueri dapat diamati secara langsung.

Secara keseluruhan, praktikum ini berhasil memberikan pemahaman yang lebih mendalam mengenai teori dan implementasi aljabar relasional dalam basis data modern, serta memperkuat dasar untuk mengembangkan keterampilan dalam pengelolaan data yang efisien menggunakan SQL.

MODUL 5

QUERY BERSYARAT

A. SOAL TUGAS PRAKTIKUM

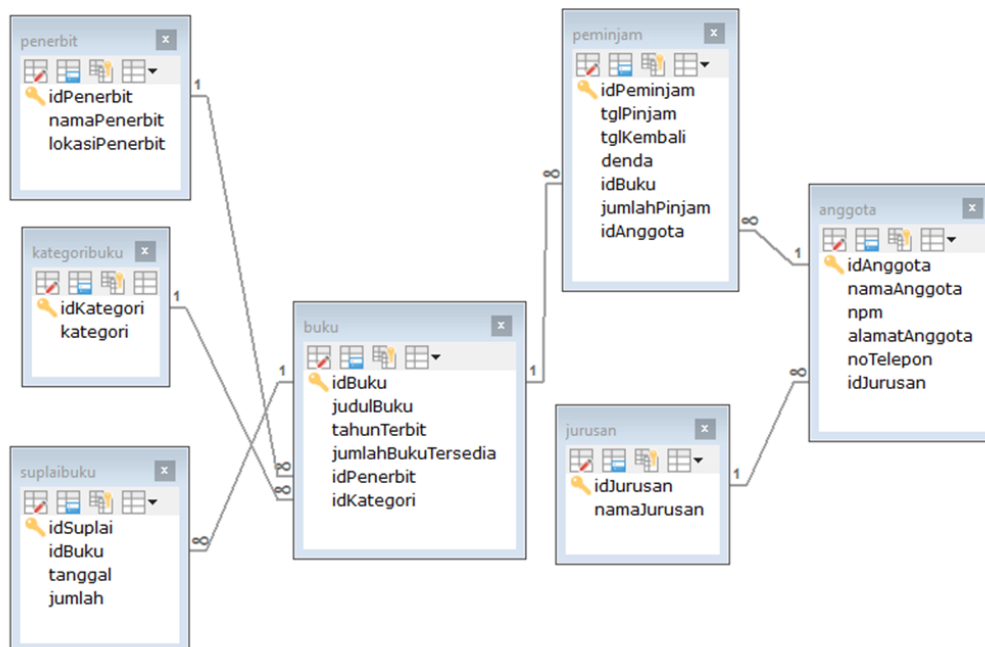
Query Bersyarat

Query bersyarat merupakan perintah query yang memiliki kriteria. Sedangkan untuk where digunakan untuk menentukan kriteria dengan menggunakan fungsi Comparison(=, <>, <, >, >= atau <=), between, in, like / not like atau is null / is not null, misalnya : `select first_name, employee_id from employees where employee_id >= 200`. Untuk kriteria having biasanya digunakan untuk menentukan kriteria yang bersifat fungsi aggregate dan harus diikuti dengan group by.

Soal Tugas Praktikum

Buatlah berbagai query SQL menggunakan klausa **WHERE** dengan operator seperti AND, OR, NOT, IS NULL, OPERATOR, LIKE, IN, BETWEEN, serta penggunaan klausa **HAVING** dengan fungsi agregat seperti AVG(), COUNT(), MAX(), MIN(), dan SUM(). Setiap query bertujuan untuk mengambil data dari database sesuai dengan kondisi atau kriteria tertentu.

Struktur CDM



B. ANALISIS MASALAH

Dalam skenario basis data yang kompleks, sering kali kita perlu mengambil data spesifik yang memenuhi kondisi-kondisi tertentu. Penggunaan klausa WHERE dan HAVING memfasilitasi penyaringan data agar hanya baris yang relevan yang dikembalikan. Berbagai operator dan fungsi agregat memungkinkan kita membuat query yang lebih fleksibel dan dinamis.

C. HASIL PRAKTIKUM

WHERE

1. Operator

a. (SQL Arithmetic Operators)

Operator aritmatika digunakan untuk melakukan operasi matematis seperti penjumlahan, pengurangan, perkalian, dan pembagian. Dalam SQL, beberapa operator aritmatika umum digunakan dalam perhitungan atau manipulasi data numerik dalam kolom tabel.

```
-- (SQL Arithmetic Operators)
SELECT * FROM buku WHERE idBuku % 2 = 0;
```

Penjelasan : Di sini digunakan operator modulus % untuk membagi nilai dalam kolom idBuku dengan 2 dan menghasilkan sisa pembagian. Baris yang memiliki nilai idBuku yang habis dibagi 2 (yaitu bilangan genap) akan ditampilkan.

```
SELECT * FROM buku WHERE jumlahBukuTersedia - idPenerbit > 3;
```

Penjelasan : Operator aritmatika - digunakan untuk mengurangi nilai dalam kolom jumlahBukuTersedia dengan idPenerbit. Kondisi ini akan menampilkan buku-buku yang hasil pengurangannya lebih besar dari 3.

b. (SQL Bitwise Operators)

Operator bitwise beroperasi pada level bit dan biasanya digunakan untuk bekerja dengan data integer. Mereka bekerja dengan merepresentasikan angka dalam biner.

```
-- (SQL Bitwise Operators)
SELECT * FROM buku WHERE idBuku & 4 = 4;
```

Penjelasan : Operator & adalah operator bitwise AND. Di sini, SQL membandingkan representasi biner dari idBuku dengan angka 4. Jika hasil dari operasi AND pada kedua angka ini menghasilkan 4, maka baris tersebut dipilih.

c. (SQL Comparison Operators)

Operator perbandingan digunakan untuk membandingkan dua nilai, seperti sama dengan (=), lebih besar (>), lebih kecil (<), lebih besar atau sama dengan (>=), dll.

```
-- (SQL Comparison Operators)
```

```
SELECT * FROM buku WHERE jumlahBukuTersedia >= 10;
```

Penjelasan : Di sini operator `>=` digunakan untuk memilih buku yang memiliki `jumlahBukuTersedia` sama dengan atau lebih besar dari 10.

```
SELECT * FROM buku WHERE idBuku < 4;
```

Penjelasan : Operator `<` digunakan untuk menampilkan buku-buku yang memiliki nilai `idBuku` kurang dari 4.

d. (SQL Compound Operators)

SQL Compound Operators memang dapat digunakan untuk memanipulasi data, namun secara umum tidak dapat langsung digunakan dalam klausa `WHERE`. Operator seperti `+=`, `-=`, `*=`, `/=`, dan `%=` lebih sering digunakan dalam klausa `UPDATE` untuk mengubah nilai kolom tertentu, bukan untuk membandingkan nilai dalam kondisi `WHERE`.

Contoh:

```
UPDATE buku SET jumlahBukuTersedia -= 1 WHERE tahunTerbit < 2018;
```

Penjelasan : Syntaks diatas error tidak bisa dijalankan karena tidak didukung dalam MySQL. Beberapa RDBMS seperti SQL Server mungkin mendukung Compound Operators, tetapi MySQL tidak. Sebagai alternatifnya, bisa menggunakan cara dibawah ini.

```
UPDATE buku SET jumlahBukuTersedia = jumlahBukuTersedia - 1 WHERE tahunTerbit < 2018;
```

Penjelasan : Untuk MySQL, kita menggunakan operasi standar untuk mengurangi nilai `jumlahBukuTersedia` sebesar 1 pada baris-baris di mana `tahunTerbit` kurang dari 2018. Di sini operator `-` digunakan untuk mengurangi nilai kolom dan hasilnya ditugaskan kembali ke kolom yang sama.

2. AND

Operator `AND` digunakan untuk menggabungkan dua atau lebih kondisi dalam klausa `WHERE`, di mana semua kondisi harus bernilai benar (`TRUE`) agar baris yang sesuai dapat dipilih.

```
-- AND
```

```
SELECT * FROM buku WHERE idBuku < 4 AND jumlahBukuTersedia >= 10;
```

Penjelasan : Query ini memilih baris dari tabel buku di mana `idBuku` kurang dari 4 dan `jumlahBukuTersedia` lebih besar atau sama dengan 10.

3. OR

Operator OR digunakan untuk menggabungkan dua atau lebih kondisi dalam klausa WHERE, di mana salah satu atau semua kondisi bisa bernilai benar (TRUE) agar baris yang sesuai dapat dipilih.

```
-- OR
SELECT * FROM buku WHERE tahunTerbit > 2020 OR idPenerbit = 10;
```

Penjelasan : Query ini memilih buku di mana tahunTerbit lebih dari 2020 atau idPenerbit sama dengan 10.

4. NOT

Operator NOT digunakan untuk membalikkan hasil dari suatu kondisi. Jika kondisi bernilai TRUE, hasilnya akan menjadi FALSE, dan sebaliknya.

```
-- Not
SELECT * FROM buku WHERE NOT judulBuku = 'Pemrograman Java';
```

Penjelasan : Query ini memilih semua buku yang bukan berjudul "Pemrograman Java".

5. IS NULL

Operator IS NULL digunakan untuk memeriksa apakah sebuah kolom memiliki nilai NULL (kosong).

```
-- Is Null
SELECT * FROM buku WHERE tahunTerbit IS NULL;
```

Penjelasan : Query ini memilih semua buku di mana nilai tahunTerbit adalah NULL atau tidak ada.

6. BETWEEN

Operator BETWEEN digunakan untuk memilih nilai dalam rentang tertentu, termasuk batas-batas rentang tersebut.

```
-- Between
SELECT * FROM buku WHERE tahunTerbit BETWEEN 2013 AND 2019;
```

Penjelasan : Query ini memilih semua buku yang diterbitkan antara tahun 2013 dan 2019, termasuk kedua tahun tersebut.

```
SELECT * FROM buku WHERE judulBuku BETWEEN 'Biologi Sel' AND 'Filsafat Barat';
```

Penjelasan : Query ini memilih semua buku yang judulnya berada dalam urutan alfabetis antara 'Biologi Sel' dan 'Filsafat Barat'.

7. LIKE

Operator LIKE digunakan dalam klausa WHERE untuk mencari pola dalam kolom teks. Dapat menggunakan wildcard seperti % untuk mencocokkan banyak karakter, atau _ untuk mencocokkan satu karakter.

```
-- Like
SELECT * FROM buku WHERE judulBuku LIKE 'p%';
```

Penjelasan : Query ini mencari semua buku yang judulnya dimulai dengan huruf 'p'.

```
SELECT * FROM buku WHERE judulBuku LIKE '%S__';
```

Penjelasan : Query ini mencari semua buku yang judulnya diakhiri dengan huruf 'S' diikuti oleh dua karakter apapun.

8. IN

Operator IN digunakan untuk menentukan apakah suatu nilai cocok dengan salah satu dari beberapa nilai yang ada dalam sebuah daftar.

```
-- In
SELECT * FROM buku WHERE tahunTerbit IN (2021, 2020, 2014);
```

Penjelasan : Query ini memilih semua buku yang tahunTerbit-nya adalah salah satu dari 2021, 2020, atau 2014.

```
SELECT * FROM buku WHERE tahunTerbit NOT IN (2021, 2020, 2014);
```

Penjelasan : Query ini memilih semua buku yang tahunTerbit-nya bukan 2021, 2020, atau 2014.

HAVING

1. AVG

Operator AVG digunakan untuk mengembalikan rata-rata (average) nilai di suatu kolom.

```
-- AVG
SELECT idBuku, AVG(denda) AS rataRataDenda
FROM Peminjam
GROUP BY idBuku
HAVING AVG(denda) > 10000;
```

Penjelasan : Query ini memilih id buku yang rata-rata denda peminjamannya lebih dari 10000.

2. COUNT

Operator COUNT digunakan untuk menghitung jumlah baris pada suatu kolom.

```
-- COUNT
SELECT idAnggota, COUNT(idBuku) AS jumlahPinjaman
FROM Peminjam
GROUP BY idAnggota
HAVING COUNT(idBuku) > 1;
```

Penjelasan : Query ini memilih id anggota yang telah meminjam lebih dari 1 buku.

3. MAX

Operator MAX digunakan untuk mengembalikan nilai terbesar di suatu kolom.

```
-- MAX
SELECT idAnggota, MAX(denda) AS dendaMax
FROM peminjam
GROUP BY idAnggota
HAVING MAX(denda) > 10000;
```

Penjelasan : Query ini memilih anggota yang pernah membayar denda dan denda maksimum lebih dari 10.000.

4. MIN

Operator MIN digunakan untuk memilih kategori buku yang memiliki stok minimum kurang dari 5 buku.

```
-- MIN
SELECT idKategori, MIN(jumlahBukuTersedia) AS stokMin
FROM buku
GROUP BY idKategori
HAVING MIN(jumlahBukuTersedia) < 5;
```

Penjelasan : Query ini memilih anggota yang pernah membayar denda dan denda maksimum lebih dari 10.000.

5. SUM

Operator SUM digunakan untuk menghitung jumlah nilai di suatu kolom.

```
-- SUM
SELECT idPeminjam, SUM(jumlahPinjam) AS totalPinjam
FROM peminjam
GROUP BY idPeminjam
HAVING SUM(jumlahPinjam) > 3;
```

Penjelasan : Query ini memilih daftar peminjam beserta total buku yang dipinjam, dengan syarat jumlah buku yang dipinjam lebih dari 3.

D. PEMBAHASAN

Pada praktikum query bersyarat terdapat beberapa kekurangan yang mungkin sering terjadi dan berikut juga solusi dari masalah tersebut :

- **AND, OR, dan NOT**
Kekurangan pada penggunaan operator AND, OR, dan NOT dapat menghasilkan tampilan yang tidak diinginkan jika penyusunannya tidak dipahami dengan baik. Untuk solusinya dapat menggunakan tanda kurung untuk mengelompokkan kondisi AND, OR, dan NOT agar MySQL dapat mengeksekusi query tersebut sesuai dengan urutan yang diinginkan.
- **IS NULL**
Kekurangan pada IS NULL terkadang pencarian data bisa tidak efisien jika kolom tidak terindeks, sehingga solusinya pada kolom data yang kosong harus memiliki indeks.
- **BETWEEN**
Kekurangan pada BETWEEN terkadang tidak mencakup nilai batasan dengan benar jika data berisi waktu atau tanggal yang memiliki ketelitian tinggi. Untuk solusinya, pastikan format data dan tipe data sudah sesuai dengan penggunaan BETWEEN. Misalnya, saat menggunakan BETWEEN dengan tipe data DATE, tentukan rentang yang benar, atau tambahkan rentang tambahan jika bekerja dengan tipe data waktu yang lebih detail.
- **Penggunaan WHERE dan HAVING secara bersamaan**
Dalam beberapa situasi, kita mungkin bingung kapan harus menggunakan WHERE dan kapan harus menggunakan HAVING. Penggunaan yang salah dapat menghasilkan tampilan yang tidak sesuai dengan yang diharapkan, sehingga untuk solusinya, kita harus lebih memahami tentang WHERE dan HAVING. WHERE digunakan untuk menyaring baris sebelum agregasi dan kondisi individual pada baris, sedangkan HAVING digunakan untuk menyaring grup setelah agregasi.

E. KESIMPULAN

Dalam praktikum query bersyarat pada MySQL, kita dapat mempelajari cara memfilter data dengan klausa WHERE, yang memungkinkan pemilihan baris berdasarkan kriteria tertentu. Operator logika seperti AND, OR, dan NOT digunakan untuk menggabungkan atau meniadakan beberapa kondisi, sedangkan IS NULL memeriksa nilai kosong, BETWEEN memilih rentang nilai, dan LIKE mencari pola dengan wildcard. Operator IN mempermudah pencocokan dengan beberapa nilai sekaligus.

Selain WHERE, kita juga dapat mempelajari klausa HAVING, yang digunakan untuk memfilter hasil agregasi setelah fungsi seperti AVG, COUNT, MAX, MIN, dan SUM diterapkan. HAVING memungkinkan pemfilteran berdasarkan nilai agregat, misalnya menampilkan transaksi yang jumlahnya lebih dari rata-rata. Dengan pemahaman ini, kita dapat lebih efektif dalam mencari dan menganalisis data di MySQL.