# MODUL PRAKTIKUM BASIS DATA

## Pertemuan 9 : Agregasi, Like, Group By, Having Basis Data pada PostgreSQL

## 1. Capaian Pembelajaran Mata Kuliah (CPMK)

Setelah mengikuti praktikum ini, mahasiswa diharapkan mampu:

1. Mengoperasikan DBMS PostgreSQL menggunakan pgAdmin.
2. Menulis query SQL dasar–menengah pada PostgreSQL.
3. Menggunakan fungsi agregasi untuk menganalisis data.
4. Menggunakan LIKE untuk pencarian data berbasis pola.
5. Mengelompokkan dan memfilter data menggunakan GROUP BY dan HAVING secara benar.

## 2. Tujuan Praktikum

Praktikum ini bertujuan agar mahasiswa:

1. Memberikan pemahaman praktis penggunaan SQL pada PostgreSQL.
2. Melatih mahasiswa melakukan pengolahan dan analisis data sederhana.
3. Membiasakan mahasiswa menggunakan pgAdmin sebagai antarmuka basis data

## 3. Dasar Teori

### 3.1 Fungsi Agregasi

Dalam konteks database, fungsi agregasi mengacu pada operasi matematika yang digunakan untuk menggabungkan beberapa baris data menjadi satu nilai tunggal. Fungsi agregasi digunakan untuk menyimpulkan atau meringkas informasi dari kumpulan data yang lebih besar. Beberapa fungsi agregasi umum termasuk:

| Fungsi Agregasi | Kegunaan |
|---|---|
| COUNT | Menghitung jumlah baris dalam satu kolom atau kelompok data. |
| SUM | Menjumlahkan nilai-nilai numerik dalam satu kolom atau kelompok data. |
| AVG | Menghitung rata-rata nilai numerik dalam satu kolom atau kelompok data. |
| MIN | Mengembalikan nilai terkecil dalam satu kolom atau kelompok data. |
| MAX | Mengembalikan nilai terbesar dalam satu kolom atau kelompok data. |

1. COUNT

```sql
SELECT COUNT(*) AS total_records FROM nama_tabel;
```

Contoh di atas menghitung jumlah baris (records) dalam tabel nama_tabel.

2. SUM

```sql
SELECT SUM(jumlah_produk) AS total_produk FROM nama_tabel;
```

Contoh di atas menjumlahkan nilai kolom jumlah_produk dari tabel nama_tabel dan memberikan hasilnya dalam kolom alias total_produk.

3. AVG

```sql
SELECT AVG(harga_produk) AS rata_rata_harga FROM nama_tabel;
```

Contoh di atas menghitung rata-rata nilai kolom harga_produk dari tabel nama_tabel dan memberikan hasilnya dalam kolom alias rata_rata_harga.

4. MIN

```sql
SELECT MIN(harga_produk) AS harga_terendah FROM nama_tabel;
```

Contoh di atas mengambil nilai terkecil dari kolom harga_produk dalam tabel nama_tabel dan memberikan hasilnya dalam kolom alias harga_terendah.

5. MAX

```sql
SELECT MAX(harga_produk) AS harga_tertinggi FROM nama_tabel;
```

Contoh di atas mengambil nilai terbesar dari kolom harga_produk dalam tabel nama_tabel dan memberikan hasilnya dalam kolom alias harga_tertinggi.

Perlu diingat bahwa dalam contoh-contoh di atas, "nama_tabel" harus diganti dengan nama tabel yang sesuai di database Anda, dan "jumlah_produk" serta "harga_produk" adalah kolom-kolom yang ada dalam tabel tersebut.

**Fungsi lain yang perlu dipahami dalam pembuatan basis data adalah sebagai berikut :**

### 1. IN

IN digunakan untuk memeriksa apakah suatu nilai termasuk dalam sekumpulan nilai tertentu.

Contoh:

```sql
SELECT * FROM mahasiswa
WHERE jurusan IN ('Informatika', 'Sistem Informasi');
```

Artinya:
Ambil data mahasiswa yang jurusannya Informatika atau Sistem Informasi.

Tidak ada batas banyaknya nilai yang bisa ada di dalam IN (...).

### 2. NOT IN

NOT IN digunakan untuk memeriksa apakah suatu nilai tidak termasuk dalam sekumpulan nilai tertentu.

Contoh:

```sql
SELECT * FROM karyawan
WHERE jabatan NOT IN ('Manager', 'Supervisor');
```

Artinya:
Ambil data karyawan yang bukan Manager dan bukan Supervisor

Nilai NULL tidak akan tampil dalam IN dan NOT IN.
Perhatikan perbedaan penggunaan OR dan AND dalam IN dan NOT IN.

### 3. BETWEEN

BETWEEN digunakan untuk memilih data yang berada di antara dua nilai batas (inklusif). Inklusif → nilai batas ikut dihitung.
Umumnya digunakan pada data numerik, tanggal, atau waktu

Contoh:

```
SELECT * FROM nilai
WHERE nilai_akhir BETWEEN 70 AND 85;
```

Artinya:
Ambil nilai yang ≥ 70 dan ≤ 85.

### 4. NOT BETWEEN

NOT BETWEEN digunakan untuk memilih data yang berada di luar rentang nilai tertentu.

Contoh:

```
SELECT * FROM produk
WHERE harga NOT BETWEEN 10000 AND 50000;
```

Artinya:
Ambil produk yang harganya kurang dari 10.000 atau lebih dari 50.000.

## 3.2. Fungsi Like, Group By, Having

### 1. LIKE

LIKE digunakan untuk melakukan pencarian pola (pattern matching) dalam kolom teks. Umumnya digunakan pada klausa WHERE.

Sintaks:

```
SELECT kolom1, kolom2
FROM nama_tabel
WHERE kolom_teks LIKE 'pola';
```

Pola:

- % = menggantikan **0 atau lebih karakter**
- _ = menggantikan **1 karakter tunggal**

Contoh:

1. Mencari data yang diawali huruf tertentu

```
SELECT * FROM mahasiswa
WHERE nama LIKE 'A%';
```

Mengambil mahasiswa yang namanya diawali huruf **A**

2. Mencari data yang mengandung kata tertentu

```
SELECT * FROM buku
WHERE judul LIKE '%database%';
```

3. LIKE tanpa memperhatikan huruf besar/kecil

```
SELECT * FROM mahasiswa
WHERE nama ILIKE '%yuni%';
```

LIKE merupakan syntak sql yang case sensitive gunakan ILIKE utuk case insensitive.

**2. GROUP BY**

GROUP BY digunakan untuk mengelompokkan data berdasarkan satu atau lebih kolom. Biasanya digunakan bersama fungsi agregat seperti SUM, AVG, COUNT, MAX, MIN.

Sintaks :

```
SELECT kolom_group, fungsi_agregat(kolom)
FROM nama_tabel
GROUP BY kolom_group;
```

Contoh:

a. Menghitung jumlah mahasiswa per jurusan

```
SELECT jurusan, COUNT(*) AS jumlah_mahasiswa
FROM mahasiswa
GROUP BY jurusan;
```

b. Menghitung rata-rata nilai per mata kuliah

```sql
SELECT mata_kuliah, AVG(nilai) AS rata_rata
FROM nilai
GROUP BY mata_kuliah;
```

### 3. HAVING

Klausa HAVING ditambahkan ke SQL karena klausa WHERE tidak dapat digunakan dengan fungsi agregat.

Fungsi agregat sering digunakan dengan klausa GROUP BY, dan dengan menambahkan HAVING kita dapat menulis kondisi seperti yang kita lakukan dengan klausa WHERE.

**Sintaks :**

```sql
SELECT kolom_group, fungsi_agregat(kolom)
FROM nama_tabel
GROUP BY kolom_group
HAVING kondisi_agregat;
```

**Contoh :**

a. Menampilkan jurusan dengan jumlah mahasiswa > 10

```sql
SELECT jurusan, COUNT(*) AS jumlah_mahasiswa
FROM mahasiswa
GROUP BY jurusan
HAVING COUNT(*) > 10;
```

b. Produk dengan total penjualan di atas 1.000.000

```sql
SELECT id_produk, SUM(total_harga) AS total_penjualan
FROM transaksi
GROUP BY id_produk
HAVING SUM(total_harga) > 1000000;
```

### 4. Kombinasi WHERE + GROUP BY + HAVING

```sql
SELECT jurusan, COUNT(*) AS jumlah_mahasiswa
FROM mahasiswa
WHERE angkatan = 2023
GROUP BY jurusan
HAVING COUNT(*) >= 5;
```

# 4. Percobaan Praktikum

## 4.1 Membuat Database

```
CREATE DATABASE praktikum_agregasi;
```

## 4.2 Membuat Tabel

### 1️⃣Tabel Mahasiswa

```
CREATE TABLE mahasiswa (
    nim VARCHAR(10) PRIMARY KEY,
    nama VARCHAR(50),
    jurusan VARCHAR(30),
    angkatan INT
);
```

### 2️⃣Tabel Nilai

```
CREATE TABLE nilai (
    id SERIAL PRIMARY KEY,
    nim VARCHAR(10),
    mata_kuliah VARCHAR(30),
    nilai INT
);
```

## 4.3 Mengisi Data

### Data Mahasiswa

```
INSERT INTO mahasiswa VALUES
('M001', 'Andi Saputra', 'Informatika', 2022),
('M002', 'Budi Hartono', 'Informatika', 2022),
('M003', 'Citra Lestari', 'Sistem Informasi', 2023),
('M004', 'Dina Ayu', 'Sistem Informasi', 2023),
('M005', 'Eko Prasetyo', 'Informatika', 2021);
```

### Data Nilai

```
INSERT INTO nilai (nim, mata_kuliah, nilai) VALUES
('M001', 'Basis Data', 85),
('M002', 'Basis Data', 75),
('M003', 'Basis Data', 90),
('M004', 'Basis Data', 70),
('M005', 'Basis Data', 65);
```

## 4.4 Percobaan dan Analisis Query

### a. Operator LIKE

Menampilkan mahasiswa yang namanya diawali huruf "A"

```
SELECT * FROM mahasiswa
WHERE nama LIKE 'A%';
```

### b. Operator IN

Menampilkan mahasiswa jurusan tertentu

```
SELECT * FROM mahasiswa
WHERE jurusan IN ('Informatika', 'Sistem Informasi');
```

### c. Operator BETWEEN

Menampilkan mahasiswa angkatan 2022–2023

```
SELECT * FROM mahasiswa
WHERE angkatan BETWEEN 2022 AND 2023;
```

### d. GROUP BY + COUNT

Menghitung jumlah mahasiswa per jurusan

```
SELECT jurusan, COUNT(*) AS jumlah_mahasiswa
FROM mahasiswa
GROUP BY jurusan;
```

### e. GROUP BY + AVG

Menghitung rata-rata nilai per mata kuliah

```sql
SELECT mata_kuliah, AVG(nilai) AS rata_rata_nilai
FROM nilai
GROUP BY mata_kuliah
```

### f. HAVING

Menampilkan mata kuliah dengan rata-rata nilai ≥ 75

```sql
SELECT mata_kuliah, AVG(nilai) AS rata_rata_nilai
FROM nilai
GROUP BY mata_kuliah
HAVING AVG(nilai) >= 75;
```

### g. Kombinasi WHERE, GROUP BY, HAVING

```sql
SELECT jurusan, COUNT(*) AS jumlah_mahasiswa
FROM mahasiswa
WHERE angkatan >= 2022
GROUP BY jurusan
HAVING COUNT(*) >= 2;
```

## 5. Latihan Praktikum

1. Dari tabel **mahasiswa**, tampilkan:
   - jurusan
   - jumlah mahasiswa

   Dengan ketentuan:
   - Nama mahasiswa **mengandung huruf "a"**
   - Angkatan **2022 ke atas**
   - Hanya tampilkan jurusan dengan **jumlah mahasiswa lebih dari 1**
2. Dari tabel **mahasiswa**, tampilkan:
   - angkatan
   - jumlah mahasiswa

   Dengan syarat:
   - Jurusan **Informatika**

- Angkatan **bukan 2021**
- Urutkan berdasarkan **jumlah mahasiswa terbanyak**

3. Dari tabel **nilai**, tampilkan:
   - mata_kuliah
   - jumlah mahasiswa yang nilainya **≥ 70**

   Hanya tampilkan:
   - Mata kuliah yang memiliki **minimal 3 mahasiswa lulus**
4. Dari tabel **nilai**, hitung:
   - Nilai **tertinggi**
   - Nilai **terendah**
   - **Rata-rata nilai**

   Untuk setiap mata kuliah, dengan ketentuan:
   - Hanya nilai **antara 60 dan 90** yang dihitung
5. Dari tabel **nilai**, tampilkan:
   - mata_kuliah
   - jumlah mahasiswa
   - rata-rata nilai

   Dengan syarat:
   - Mata kuliah **Basis Data**
   - Nilai yang dihitung **≥ 65**
   - Rata-rata nilai **≥ 75**
6. Dari tabel **mahasiswa**, tampilkan:
   - jurusan
   - angkatan
   - jumlah mahasiswa

   Dengan ketentuan:
   - Nama mahasiswa **diawali huruf A atau B**
   - Tampilkan hanya kelompok data yang memiliki **lebih dari 1 mahasiswa**
7. Dari tabel **nilai**, tampilkan:
   - nim
   - jumlah mata kuliah yang diambil

- nilai rata-rata

Dengan ketentuan:

- Hanya nilai **≥ 60**
- Mahasiswa mengambil **lebih dari 1 mata kuliah**

8. Dari tabel **nilai**, tampilkan:
   - mata_kuliah
   - jumlah mahasiswa
   - selisih nilai tertinggi dan terendah

   Dengan ketentuan:
   - Mata kuliah memiliki **minimal 3 mahasiswa**
   - Selisih nilai **≥ 10**
9. Dari tabel **mahasiswa**, tampilkan:
   - jurusan
   - jumlah mahasiswa

   Dengan ketentuan:
   - Angkatan **antara 2022 dan 2024**
   - Nama **tidak mengandung huruf "o"**
   - Jumlah mahasiswa **≥ 2**
10. Dari tabel **nilai**, tampilkan:
    - mata_kuliah
    - jumlah mahasiswa
    - rata-rata nilai

    Dengan ketentuan:
    1. Hanya nilai **antara 65 dan 85**
    2. Mata kuliah memiliki **lebih dari 2 mahasiswa**
    3. Urutkan berdasarkan **rata-rata nilai tertinggi**