LAPORAN PRAKTIKUM BASIS DATA RC

Muhammad Yusuf
122140193

Latihan

Teori Dasar

Teori dasar agregasi dalam MySQL melibatkan penggunaan fungsi agregasi seperti SUM, AVG, COUNT, MIN, dan MAX untuk melakukan operasi perhitungan pada data yang ada dalam tabel. Fungsi-fungsi ini digunakan untuk mengumpulkan informasi statistik tentang data yang diproses. Berikut adalah penjelasan lebih lanjut tentang teori dasar agregasi dalam MySQL:

1. Fungsi Agregasi
   Fungsi-fungsi agregasi memungkinkan kita untuk melakukan operasi perhitungan seperti menjumlahkan nilai-nilai dalam suatu kolom, menghitung rata-rata, mengambil nilai maksimum atau minimum, dan menghitung jumlah baris dalam suatu kumpulan data.
2. Penggunaan dalam SELECT
   Fungsi agregasi dapat digunakan dalam klausa SELECT untuk mengambil nilai-nilai agregat dari suatu kumpulan data. Misalnya, `SELECT SUM(total_harga) FROM penjualan` akan menghitung total harga dari semua transaksi penjualan.
3. Klausa GROUP BY
   Klausa GROUP BY digunakan bersamaan dengan fungsi agregasi untuk mengelompokkan data berdasarkan nilai tertentu, seperti `GROUP BY kategori_produk` untuk mengelompokkan penjualan berdasarkan kategori produk.
4. Hanya dalam SELECT
   Fungsi-fungsi agregasi hanya dapat digunakan di dalam klausa SELECT, kecuali COUNT, yang juga dapat digunakan dalam klausa WHERE atau HAVING untuk menghitung jumlah baris yang memenuhi suatu kondisi.
5. Penanganan Nilai NULL
   Fungsi agregasi secara default mengabaikan nilai NULL dalam perhitungan. Namun, kita dapat menggunakan IFNULL atau COALESCE untuk mengatasi nilai NULL jika diperlukan.
6. Klausa HAVING
   Klausa HAVING digunakan bersamaan dengan GROUP BY untuk memberikan kondisi filter terhadap hasil agregasi. Misalnya, `HAVING COUNT(*) > 10` akan memfilter grup yang memiliki lebih dari 10 baris.
7. Urutan Operasi
   Urutan operasi dalam klausa SELECT adalah FROM, WHERE, GROUP BY, HAVING, SELECT, ORDER BY. Ini berarti fungsi agregasi diterapkan setelah klausa WHERE dan sebelum klausa ORDER BY.
8. Penggunaan Aliases
   Kita dapat menggunakan aliases untuk memberikan nama yang lebih deskriptif pada hasil fungsi agregasi, misalnya, `SELECT SUM(total_harga) AS total_penjualan FROM penjualan`.

Dengan memahami teori dasar agregasi dalam MySQL, kita dapat membuat query yang kompleks untuk menganalisis data secara efektif dan mendapatkan informasi yang dibutuhkan dari database. Namun, perlu diingat untuk memperhatikan kinerja query agar tetap efisien, terutama ketika menggabungkan fungsi agregasi dengan subquery atau operasi lainnya.

Pembahasan

1. Buatlah database dengan nama ”Akademik”

```
MariaDB [(none)]> create database Akademik;
Query OK, 1 row affected (0.002 sec)
```

2. Buatlah tabel dengan nama ”mahasiswa” dengan struktur tabel sebagai berikut

Field	Type	Null	Key	Default	Extra
nim	char(3)	NO		NULL	
kodemk	varchar(5)	NO		NULL	
thnakademik	char(9)	NO		NULL	
nilai	char(1)	YES		NULL	
bobot	int(2)	NO		NULL	

```
MariaDB [(none)]> use akademik;
Database changed
MariaDB [akademik]> create table mahasiswa(
    ->      nim char(3),
    ->      kodemk varchar(5),
    ->      thnakademik char(9),
    ->      nilai char(1),
    ->      bobot int(2)
    -> );
Query OK, 0 rows affected (0.010 sec)
```

3. Masukan data berikut ke dalam tabel ”mahasiswa”

nim	kodemk	thnakademik	nilai	bobot
123	SMBD2	2020/2021	C	4
123	SMBD2	2021/2022	A	4
123	SIBW	2021/2022	A	4
123	DMEP	2021/2022	B	2
456	DMEP	2021/2022	A	2
456	SIBW	2019/2020	C	4
456	SIBW	2020/2021	C	4
456	SIBW	2021/2022	A	4
789	SMBD2	2017/2018	D	4
789	SMBD2	2018/2019	C	4
789	SMBD2	2019/2020	C	4
789	SMBD2	2020/2021	A	4

```
MariaDB [akademik]> insert into mahasiswa values
    -> ('123', 'SMBD2', '2020/2021', 'C', 4),
    -> ('123', 'SMBD2', '2021/2022', 'A', 4),
    -> ('123', 'SIBW', '2021/2022', 'A', 4),
    -> ('123', 'DMEP', '2021/2022', 'B', 2),
    -> ('456', 'DMEP', '2021/2022', 'A', 2),
    -> ('456', 'SIBW', '2019/2020', 'C', 4),
    -> ('456', 'SIBW', '2020/2021', 'C', 4),
    -> ('456', 'SIBW', '2021/2022', 'A', 4),
    -> ('789', 'SMBD2', '2017/2018', 'D', 4),
    -> ('789', 'SMBD2', '2018/2019', 'C', 4),
    -> ('789', 'SMBD2', '2019/2020', 'C', 4),
    -> ('789', 'SMBD2', '2020/2021', 'A', 4);
Query OK, 12 rows affected (0.003 sec)
Records: 12  Duplicates: 0  Warnings: 0
```

4. Tampilkan nilai terbaik yang didapatkan oleh seorang mahasiswa mata kuliah tertentu.

```
MariaDB [akademik]> SELECT nim, kodemk, MAX(bobot) AS nilai_terbaik FROM mah
asiswa GROUP BY nim, kodemk;
+------+--------+---------------+
| nim  | kodemk | nilai_terbaik |
+------+--------+---------------+
| 123  | DMEP   |             2 |
| 123  | SIBW   |             4 |
| 123  | SMBD2  |             4 |
| 456  | DMEP   |             2 |
| 456  | SIBW   |             4 |
| 789  | SMBD2  |             4 |
+------+--------+---------------+
6 rows in set (0.001 sec)
```

5. Tampilkan mata kuliah berserta nilai yang terburuk yang pernah didapatkan oleh mahasiswa dengan nim 123

```
MariaDB [akademik]> SELECT kodemk, MIN(bobot) AS nilai_terburuk FROM mahasis
wa WHERE nim = '123' GROUP BY kodemk;
+--------+----------------+
| kodemk | nilai_terburuk |
+--------+----------------+
| DMEP   |              2 |
| SIBW   |              4 |
| SMBD2  |              4 |
+--------+----------------+
3 rows in set (0.001 sec)
```

6. Tampilkan jumlah cacah nilai yang pernah diberikan untuk matakuliah tertentu

```
MariaDB [akademik]> SELECT kodemk, COUNT(nilai) AS jumlah_nilai FROM mahasis
wa GROUP BY kodemk;
+--------+--------------+
| kodemk | jumlah_nilai |
+--------+--------------+
| DMEP   |            2 |
| SIBW   |            4 |
| SMBD2  |            6 |
+--------+--------------+
3 rows in set (0.001 sec)
```

7. Tampilkan seluruh mahasiswa yang pernah mengulang sebuah mata kuliah

```
MariaDB [akademik]> SELECT DISTINCT nim
    -> FROM mahasiswa
    -> WHERE kodemk IN (
    ->     SELECT kodemk
    ->     FROM mahasiswa
    ->     GROUP BY kodemk
    ->     HAVING COUNT(DISTINCT thnakademik) > 1
    -> );
+------+
| nim  |
+------+
| 123  |
| 456  |
| 789  |
+------+
3 rows in set (0.001 sec)
```

8. Tampilkan semua tabel dimana fieldnya terdiri dari nim, kodemk, thnakademik, nilai dimana nilai lebih kecil dari C.

```
MariaDB [akademik]> SELECT *
    -> FROM mahasiswa
    -> WHERE nilai < 'C';
+------+--------+-------------+-------+-------+
| nim  | kodemk | thnakademik | nilai | bobot |
+------+--------+-------------+-------+-------+
| 123  | SMBD2  | 2021/2022   | A     |     4 |
| 123  | SIBW   | 2021/2022   | A     |     4 |
| 123  | DMEP   | 2021/2022   | B     |     2 |
| 456  | DMEP   | 2021/2022   | A     |     2 |
| 456  | SIBW   | 2021/2022   | A     |     4 |
| 789  | SMBD2  | 2020/2021   | A     |     4 |
+------+--------+-------------+-------+-------+
6 rows in set (0.001 sec)

MariaDB [akademik]> SELECT *
    -> FROM mahasiswa
    -> WHERE nilai > 'C';
+------+--------+-------------+-------+-------+
| nim  | kodemk | thnakademik | nilai | bobot |
+------+--------+-------------+-------+-------+
| 789  | SMBD2  | 2017/2018   | D     |     4 |
+------+--------+-------------+-------+-------+
1 row in set (0.001 sec)
```

Analisis & Kesimpulan

Setelah praktikum tentang agregasi dalam MySQL, saya memperoleh pemahaman yang kuat tentang bagaimana menghitung statistik dan menganalisis data dengan efektif. Penggunaan fungsi agregasi seperti SUM, AVG, COUNT, dan lainnya memberikan kemudahan dalam melakukan perhitungan yang kompleks, seperti menghitung total nilai, rata-rata, atau mengelompokkan data berdasarkan kriteria tertentu. Selain itu, penggunaan klausa GROUP BY dan HAVING memberikan fleksibilitas dalam mengatur hasil agregasi berdasarkan kelompok data yang berbeda, sehingga memungkinkan analisis yang lebih mendalam.

Namun, pentingnya juga untuk memperhatikan pengoptimalan kinerja saat menggunakan fungsi agregasi, terutama dalam konteks database besar. Memilih indeks yang tepat, merancang struktur query yang efisien, dan menghindari penggunaan subquery yang berlebihan dapat membantu meningkatkan kinerja query secara keseluruhan. Dengan demikian, praktikum ini tidak hanya meningkatkan pemahaman tentang agregasi dalam MySQL, tetapi juga memberikan wawasan tentang pentingnya desain query yang efisien untuk mengoptimalkan analisis data.