**BAB V**

**AGREGASI SQL DAN VIEW**

5.1 Bahasan dan Sasaran

5.1.1 Bahasan

- Agregasi sql yang diperlukan dalam perhitungan data

- View yang merupakan tabel bayangan yang berisi query baik dari satu tabel atau berbagai tabel

5.1.1 Sasaran

1. Mahasiswa memahami operator agregasi sql sehingga mempermudah dalam hal perhitungan data.

2. Mahasiswa memahami cara penggunaan view.

5.2 Materi

5.2.1 AGREGATE OPERATOR

Fungsi aggregate atau disebut fungsi ringkasan digunakan untuk melakukan penghitungan menjadi sebuah nilai dari beberapa nilai input. *Aggregate* dapat digabungkan dengan sebuah parameter seperti WHERE untuk menghasilkan suatu hasil yang lebih kompleks lagi. Adapun fungsi *agregate* yang disediakan oleh PostgreSQL dapat dilihat pada tabel berikut :

|  |  |
| --- | --- |
| Agregate | Keterangan |
| COUNT(\*) | Menghitung jumlah baris |
| SUM(NAMA KOLOM) | Menghitung penjumlahan data |
| MAX(NAMA\_KOLOM) | Mencari nilai terbesar |
| MIN(NAMA KOLOM) | Mencari nilai terkecil |
| AVG(NAMA KOLOM) | Mencari nilai rata-rata |

Berikut contoh aggregate query dari suatu tabel pegawai :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id\_peg | Nama\_peg | Alamat\_peg | Telp\_peg | Jabatan\_peg |
| 1 | Hendro | Solo | 081223300 | Teknisi |
| 2 | Tika | Semarang | 0897735357 | Sekretaris |
| 3 | Wijaya | Jogjakarta | 0865433225 | Kepala |
| 4 | Dodi | Banyuwangi | 076544677 | Staf |

Untuk pencarian banyaknya pegawai kita bisa menggunakan query berikut :

***select count(\*) from pegawai;***

hasil : 4

untuk pencarian nilai terbesar berdasarkan ID :

***select max(Id\_peg) from pegawai;***

hasi : 4

untuk pencarian nilai terkecil :

***select min(Id\_peg) from pegawai;***

hasil : 1

untuk pencarian rata-rata :

***select avg(Id\_peg) from pegawai;***

hasil : 2.5000

5.2.2 GROUP BY

Group By merupakan fungsi yang digunakan untuk melakukan pengelompokan dari perintah SELECT. Group by seringkali diperlukan untuk menjalankan *agregate* menjadi sebuah kelompok dari hasil Query. Berikut struktur SQL untuk penampilan data :

**select nama\_kolom from nama\_tabel group by nama\_kolom;**

Contoh :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Kode\_buk | Judul\_buk | Pengarang\_buk | Penerbit\_buk | Tahun\_buk | Resensi\_buk | Harga\_buk |
| 11 | Harry Potter | JK Rowling | British | 2002 | Fiksi | 50000 |
| 12 | Sistem Basis Data | Abdul Kadir | Andi | 2002 | Teks | 30000 |
| 13 | Pemrograman | Abdul Kadir | Andi | 2004 | Teks | 60000 |

* Untuk menampilkan informasi nama pengarang :

***Select pengarang\_buk from buku group by pengarang\_buk;***

Hasil :

|  |
| --- |
| Pengarang |
| JK Rowling |
| Abdul Kadir |

Pada hasil query terlihat pengarang muncul hanya sekali.

- Untuk menampilkan informasi nama pengarang beserta jumlah buku yang dikarang :

***Select pengarang\_buk, count(\*) from buku group by pengarang\_buk;***

Hasil :

|  |  |
| --- | --- |
| Pengarang | Count(\*) |
| JK Rowling | 1 |
| Abdul Kadir | 2 |

- Untuk menampilkan informasi buku tiap tahunnya :

***Select tahun\_buk, count(\*) from buku group by tahun\_buk;***

Hasil :

|  |  |
| --- | --- |
| Tahun | Count(\*) |
| 2004 | 1 |
| 2002 | 2 |

- Untu menampilkan informasi jumlah total uang tiap tahunnya :

***Select tahun\_buk, sum(harga\_buk) as total from buku group by tahun\_buk;***

Hasil :

|  |  |
| --- | --- |
| Tahun | Total |
| 2004 | 60000 |
| 2002 | 80000 |

5.2.3 HAVING

Pemakaian HAVING terkait dengan GROUP BY, kegunaanya adalah untuk menentukan kondisi bagi GROUP BY, dimana kelompok yang memenuhi kondisi saja yang akan di hasilkan.

Berikut Struktur yang digunakan :

Kita akan menggunakan tabel “pesanan”

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Tanggal Order** | **Harga Order** | **Pelanggan** |
| 1 | 2011/11/12 | 1000 | Ratih |
| 2 | 2011/10/23 | 1600 | Anita |
| 3 | 2011/09/02 | 700 | Ratih |
| 4 | 2011/09/03 | 300 | Ratih |
| 5 | 2011/08/30 | 2000 | Teguh |
| 6 | 2011/10/04 | 100 | Anita |

* Sekarang jika kita ingin mencari pelanggan yang memiliki total order kurang dari 2000.

Maka, kita dapat menggunakan statement :

**select pelanggan, sum (hargaorder) from pesanan group by pelanggan having sum (hargaorder) <2000;**

Maka, hasilnya :

|  |  |
| --- | --- |
| Pelanggan | SUM(HargaOrder) |
| Anita | 1700 |

5.2.4 CASE

Meskipun SQL bukan merupakan sebuah prosedur bahasa perograman, namun dalam prosesnya dapat dengan bebas mengontrol data yang kembali dari *query*. Kata WHERE menggunakan perbandingan untuk mengontrol pemilihan data, sedangkan *CASE* perbandingan dalam bentuk output kolom. Jadi intinya penggunaan *CASE* akan membentuk output tersendiri berupa sebuah kolom baru dengan data dari operasi yang di dalamnya.Struktur didalam select seperti berikut :

CASE WHEN *condition* THEN *result*

[WHEN ...]

[ELSE *result*] END

Berikut contoh query penggunaan case, penentuan umur jika umurnya dibawah 1986-01-01 dianggap dewasa dan lebih dari itu dianggap remaja :

**select txt\_namadepan, txt\_namaakhir, dt\_tgllahir, case when dt\_tgllahir < '1986-01-01' then 'dewasa' else 'balita' end as umur from pegawai ;**

Hasil :

|  |  |  |  |
| --- | --- | --- | --- |
| txt\_NamaDepan | txt\_NamaAkhir | dt\_TglLahir | Umur |
| Andhie | Lala | 1960-08-08 | Dewasa |
| Ade | Fajar | 1986-11-01 | Balita |
| Panuju | Sasongko | 1970-09-12 | Dewasa |
| Dudy | Rudianto | 1973-12-11 | Dewasa |
| Ana | Hidayati | 1988-10-01 | Balita |

5.2.5 VIEW

Views dapat juga disebut tabel bayangan tetapi bukan *temporary table*, bukan juga merupakan sebuah tabel yang asli. Suatu view adalah suatu relasi virtual yang tidak perlu ada database tetapi dapat diproduksi atas permintaan oleh pemakai tertentu, pada ketika permintaan*.* Satu lagi kelebihan yang dimiliki oleh view yaitu dapat menyimpan perintah *query*, dan dapat mewakili sebuah subset dari tabel asli dan memilih kolom atau *row* tertentu dari tabel biasa.

**create view nama\_tabel\_view as query;**

Catatan : Query diatas merupakan query untuk menampilkan data menggunakan query sql select.

Berikut adalah tabel contoh kasus penggunaan VIEW :

Kita akan menggunakan tabel “pesanan”

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Tanggal Order** | **Harga Order** | **Pelanggan** |
| 1 | 2011/11/12 | 1000 | Ratih |
| 2 | 2011/10/23 | 1600 | Anita |
| 3 | 2011/09/02 | 700 | Ratih |
| 4 | 2011/09/03 | 300 | Ratih |
| 5 | 2011/08/30 | 2000 | Teguh |
| 6 | 2011/10/04 | 100 | Anita |

* Kita akan membuat view dari tabel diatas dengan ketentuan harga dikumpulkan berdasarkan nama pelanggannya. Sebagai berikut :

***create view total\_pelanggan as select pelanggan, sum (hargaorder) from pesanan group by pelanggan;***

Untuk melihat hasil kita bisa melakukan query *select* sebagai berikut :

***Select \* from total\_pelanggan;***

Hasil :

|  |  |
| --- | --- |
| Pelanggan | SUM(HargaOrder) |
| Anita | 1700 |
| Ratih | 2000 |
| Teguh | 2000 |