DQLab

live Class

19 October 2023

Konsep dasar SQL : DDL

Sesi 2-Bootcamp Data Analyst with SQL and Python using Google Platform

Memahami perintah DDL

Data Definition Language (DDL) merupakan perintah SQL untuk membuat, merubah atau menghapus struktur database.

Perintah DDL antara lain

- CREATE -> Membuat database
- DROP -> Menghapus table/database

Perintah CREATE

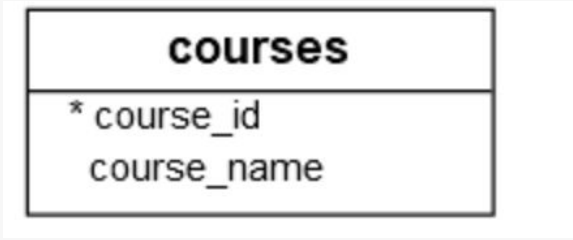
Database berisi table untuk menyimpan entitas. Table terdiri dari field (kolom) dan record (baris data). Perintah CREATE digunakan untuk membuat table.

```
CREATE TABLE table_name(
     column_name_1 data_type default value column_constraint,
     column_name_2 data_type default value column_constraint,
     ...,
     table_constraint
);
```

Contoh SQL CREATE TABLE

Berikut adalah pernyataan yang membuat tabel courses:

```
CREATE TABLE courses (
    course_id INT AUTO_INCREMENT PRIMARY KEY,
    course_name VARCHAR(50) NOT NULL
);
```



course_id adalah kolom primary key dari tabel courses. Setiap tabel memiliki satu dan hanya satu primary key yang secara unik mengidentifikasi setiap baris dalam tabel. Menentukan primary key adalah praktik yang baik untuk setiap tabel.

Tipe data dari course_id adalah integer yang ditandai dengan kata kunci INT. Selain itu, nilai kolom course_id diatur sebagai AUTO_INCREMENT. Ini berarti ketika Anda menyisipkan baris baru ke dalam tabel courses tanpa memberikan nilai untuk kolom course_id, sistem database akan menghasilkan nilai integer untuk kolom tersebut.

course_name menyimpan nama-nama kursus. Tipe data kolom ini adalah string karakter (VARCHAR) dengan panjang maksimum 50. NOT NULL memastikan bahwa tidak ada nilai NULL disimpan dalam kolom course_name.

Perintah DROP TABLE

Untuk menghapus tabel yang sudah ada, Anda menentukan nama tabel setelah klausa DROP TABLE. Jika tabel yang dihapus tidak ada, sistem database akan mengeluarkan pesan error.

Berikut adalah contoh sintaks dari pernyataan DROP TABLE:

```
DROP TABLE [IF EXISTS] table_name;
```

DQLab

live Class

Query WHERE, SORT, AGREGASI, dan HAVING

19 October 2023

Sesi 3- Bootcamp Data Analyst with SQL and Python using Google Platform

DQLab

SQL SELECT - memilih data dari kolom-kolom tertentu

Untuk memilih data dari kolom-kolom tertentu, Anda dapat menentukan daftar kolom setelah klausa SELECT dalam pernyataan SELECT.

Sebagai contoh, berikut adalah contoh untuk memilih data dari kolom employee id, first name, last name, dan hire date dari semua baris dalam tabel employees:

Pernyataan ini akan mengembalikan data yang dipilih hanya dari kolom-kolom yang disebutkan, yaitu employee_id, first_name, last_name, dan hire_date, dari semua baris dalam tabel employees.

```
SELECT
    employee_id,
    first_name,
    last_name,
    hire_date
FROM
    employees;
```

```
+-------------+-------------+-------------+-----------+
| employee_id | first_name  | last_name   | hire_date |
+-------------+-------------+-------------+-----------+
|         100 | Steven      | King        | 1987-06-17 |
|         101 | Neena       | Kochhar     | 1989-09-21 |
|         102 | Lex         | De Haan     | 1993-01-13 |
|         103 | Alexander   | Hunold      | 1990-01-03 |
|         104 | Bruce       | Ernst       | 1991-05-21 |
|         105 | David       | Austin      | 1997-06-25 |
|         106 | Valli       | Pataballa   | 1998-02-05 |
|         107 | Diana       | Lorentz     | 1999-02-07 |
|         108 | Nancy       | Greenberg   | 1994-08-17 |
|         109 | Daniel      | Faviet      | 1994-08-16 |
|         110 | John        | Chen        | 1997-09-28 |
...
```

SQL SELECT – Kalkulasi sederhana

Contoh berikut menggunakan pernyataan SELECT untuk mendapatkan first name, last name, salary, dan new salary:

Ekspresi salary * 1.05 menambahkan 5% ke gaji setiap karyawan. Secara default, SQL menggunakan ekspresi sebagai judul kolom

Pernyataan ini akan mengembalikan data yang terdiri dari first name, last name, salary, dan new salary. Kolom new_salary akan menampilkan gaji
baru yang diperoleh dengan menambahkan 5% ke gaji yang ada.

```
SELECT
    first_name,
    last_name,
    salary,
    salary * 1.05
FROM
    employees;
```

```
+-------------+-------------+----------+---------------+
| first_name  | last_name   | salary   | salary * 1.05 |
+-------------+-------------+----------+---------------+
| Steven      | King        | 24000.00 |    25200.0000 |
| Neena       | Kochhar     | 17000.00 |    17850.0000 |
| Lex         | De Haan     | 17000.00 |    17850.0000 |
| Alexander   | Hunold      |  9000.00 |     9450.0000 |
| Bruce       | Ernst       |  6000.00 |     6300.0000 |
| David       | Austin      |  4800.00 |     5040.0000 |
| Valli       | Pataballa   |  4800.00 |     5040.0000 |
| Diana       | Lorentz     |  4200.00 |     4410.0000 |
| Nancy       | Greenberg   | 12000.00 |    12600.0000 |
```

Sort

Klausa ORDER BY adalah opsional dalam pernyataan SELECT. Klausa ORDER BY memungkinkan Anda mengurutkan baris-baris yang dikembalikan oleh klausa SELECT berdasarkan satu atau lebih ekspresi pengurutan secara menaik (ascending) atau menurun (descending).

Berikut adalah sintaks dari klausa ORDER BY:

```
SELECT
    select_list
FROM
    table_name
ORDER BY
    sort_expression [ASC | DESC];
```

Pertama, letakkan klausa ORDER BY setelah klausa FROM. Database akan mengevaluasi pernyataan SELECT dengan klausa ORDER BY dalam urutan berikut: FROM > SELECT > ORDER BY.

Kedua, tentukan ekspresi pengurutan setelah klausa ORDER BY. Ekspresi pengurutan menentukan kriteria pengurutan.

Ketiga, gunakan opsi ASC untuk mengurutkan hasil set dengan ekspresi pengurutan secara menaik (ascending) dan DESC untuk mengurutkan hasil set dengan ekspresi pengurutan secara menurun (descending).

Aggregate Functions

Fungsi agregat SQL melakukan perhitungan pada sekumpulan nilai dan mengembalikan satu nilai tunggal. Sebagai contoh, fungsi rata-rata (AVG) mengambil daftar nilai dan mengembalikan nilai rata-ratanya.

Karena fungsi agregat beroperasi pada sekumpulan nilai, fungsi tersebut sering digunakan dengan klausa GROUP BY dalam pernyataan SELECT. Klausa GROUP BY membagi hasil set ke dalam kelompok-kelompok nilai, dan fungsi agregat mengembalikan satu nilai untuk setiap kelompok.

Berikut ini adalah contoh penggunaan fungsi agregat dengan klausa GROUP BY:

```
SELECT c1, aggregate_function(c2)
FROM table
GROUP BY c1;
```

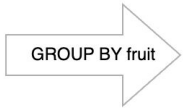
Berikut adalah beberapa fungsi agregat SQL yang umum digunakan:

- AVG() - mengembalikan rata-rata dari sebuah set nilai.
- COUNT() - mengembalikan jumlah item dalam sebuah set.
- MAX() - mengembalikan nilai maksimum dalam sebuah set.
- MIN() - mengembalikan nilai minimum dalam sebuah set.
- SUM() - mengembalikan jumlah dari semua nilai atau nilai yang berbeda dalam sebuah set.

Ilustrasi Aggregate Functions (1)

Gambar berikut mengilustrasikan cara kerja klausa GROUP BY:

id	fruit
1	Apple
2	Orange
3	Apple
4	Banana
5	Orange

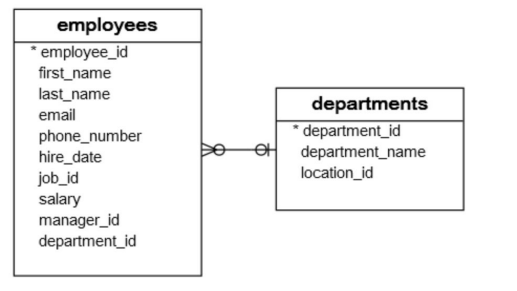


fruit
Apple
Banana
Orange

Tabel di sisi kiri memiliki dua kolom id dan buah. Saat Anda menerapkan klausa GROUP BY ke kolom buah, ini mengembalikan kumpulan hasil yang menyertakan nilai unik dari kolom buah:

```
SELECT
    fruit
FROM
    sample_table
GROUP BY
    fruit;
```

Distinct



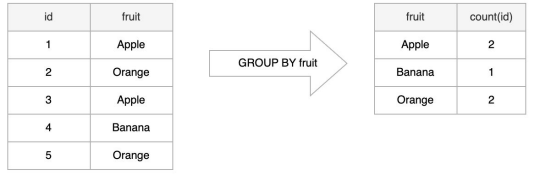
Kolom id_departemen pada tabel karyawan memiliki 40 baris, termasuk nilai id_departemen duplikat. Namun, GROUP BY mengelompokkan nilai-nilai ini ke dalam beberapa grup.
Tanpa fungsi agregat, GROUP BY berlaku seperti kata kunci DISTINCT:

```
SELECT
   DISTINCT department_id
FROM
   employees
ORDER BY
   department_id;
```

Ilustrasi Aggregate Functions (2)

Dalam praktiknya, Kita sering menggunakan klausa GROUP BY dengan fungsi agregat seperti MIN, MAX, AVG, SUM, atau COUNT untuk menghitung ukuran yang menyediakan informasi untuk setiap grup.

Misalnya, berikut ini mengilustrasikan cara kerja klausa GROUP BY dengan fungsi agregat COUNT:



Dalam contoh ini, kami mengelompokkan baris berdasarkan nilai kolom buah dan menerapkan fungsi COUNT ke kolom id. Kumpulan hasil mencakup nilai unik dari kolom buah dan jumlah baris yang sesuai.

```
SELECT
   fruit, COUNT(id)
FROM
   sample_table
GROUP BY
   fruit;
```

Fungsi Count()

Fungsi COUNT() mengembalikan jumlah item dalam satu set.

Misalnya, contoh berikut menggunakan fungsi COUNT(*) untuk mengembalikan jumlah pegawai setiap departemen:

```
SELECT
    department_name, COUNT(*) headcount
FROM
    employees
        INNER JOIN
    departments USING (department_id)
GROUP BY department_name
ORDER BY department_name;
```

department_name	headcount
Accounting	2
Administration	1
Executive	3
Finance	6
Human Resources	1
IT	5
Marketing	2
Public Relations	1
Purchasing	6
Sales	6
Shipping	7

Fungsi AVG()

Fungsi SQL AVG adalah fungsi agregat yang menghitung nilai rata-rata dari suatu kumpulan. Berikut ini mengilustrasikan sintaks dari fungsi SQL AVG:

```
AVG([ALL|DISTINCT] expression)
```

Jika kita menggunakan kata kunci ALL, fungsi AVG mengambil semua nilai dalam perhitungan. Secara default, fungsi AVG menggunakan ALL apakah kita menentukannya atau tidak.
Jika kita menetapkan kata kunci DISTINCT secara eksplisit, fungsi AVG hanya akan mengambil nilai unik dalam perhitungan.
Misalnya, kita memiliki set (1,2,3,3,4) dan menerapkan AVG(ALL) ke set ini, fungsi AVG akan melakukan perhitungan berikut:

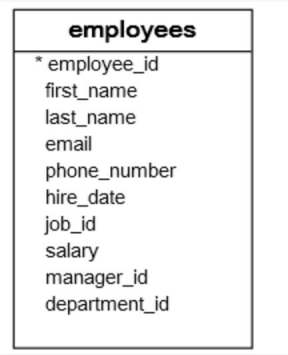
```
(1+2+3+3+4)/5 = 2.6
```

Namun, AVG(DISTINCT) akan memproses sebagai berikut:

```
(1+2+3+4)/4 = 2.5
```

Fungsi AVG()

kita akan menggunakan tabel karyawan di database sampel untuk mendemonstrasikan cara kerja fungsi SQL AVG. Gambar berikut mengilustrasikan struktur tabel karyawan:



Untuk menghitung gaji rata-rata semua karyawan, Kit menerapkan fungsi AVG ke kolom gaji sebagai berikut:

```
SELECT
    AVG(salary)
FROM
    employees;
```

	AVG(salary)
▶	8060.000000

Fungsi MIN()

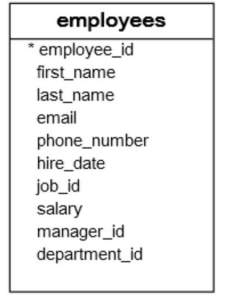
Fungsi SQL MIN mengembalikan nilai minimum dalam satu set nilai. Berikut ini menunjukkan sintaks dari fungsi MIN.

```
MIN(expression)
```

Seperti fungsi MAX, fungsi MIN juga mengabaikan nilai NULL dan opsi DISTINCT tidak berlaku untuk fungsi MIN.

Contoh Fungsi MIN()

Kita akan menggunakan tabel karyawan untuk mendemonstrasikan fungsionalitas fungsi MIN.



Untuk mencari gaji karyawan terendah (minimum), Anda menerapkan fungsi MIN ke kolom gaji pada tabel karyawan.

```
SELECT
    MIN(salary)
FROM
    employees;
```

	MIN(salary)
▶	2500.00

Fungsi MAX()

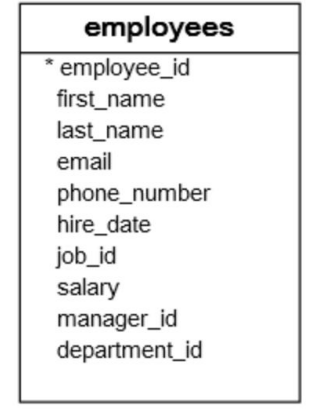
SQL menyediakan fungsi MAX yang memungkinkan Anda menemukan nilai maksimum dalam sekumpulan nilai. Berikut ini ilustrasi sintaks dari fungsi MAX.

```
MAX(expression)
```

Fungsi MAX mengabaikan nilai NULL.
Berbeda dengan fungsi SUM, COUNT, dan AVG, opsi DISTINCT tidak berlaku untuk fungsi MAX

Contoh Fungsi MAX()

Kita akan menggunakan tabel karyawan untuk mendemonstrasikan cara kerja fungsi MAX.



Pernyataan SELECT berikut mengembalikan gaji karyawan tertinggi (maksimum) dalam tabel karyawan.

```
SELECT
    MAX(salary)
FROM
    employees;
```

Fungsi SUM()

Fungsi SUM() mengembalikan jumlah semua nilai.

Misalnya, pernyataan berikut mengembalikan total gaji semua karyawan di setiap departemen:

```
SELECT
    department_id, SUM(salary)
FROM
    employees
GROUP BY department_id;
```

department_name	SUM(salary)
Accounting	20300.00
Administration	4400.00
Executive	58000.00
Finance	51600.00
Human Resources	6500.00
IT	28800.00
Marketing	19000.00
Public Relations	10000.00
Purchasing	24900.00
Sales	57700.00
Shipping	41200.00

SQL HAVING

Untuk menentukan kondisi grup, Anda menggunakan klausa HAVING.

Klausa HAVING sering digunakan dengan klausa GROUP BY dalam pernyataan SELECT. Jika Anda menggunakan klausa HAVING tanpa klausa GROUP BY, klausa HAVING berperilaku seperti klausa WHERE.

Berikut ini mengilustrasikan sintaks dari klausa HAVING:

```
SELECT
        column1,
        column2,
        AGGREGATE_FUNCTION (column3)
FROM
        table1
GROUP BY
        column1,
        column2
HAVING
        group_condition;
```

SQL HAVING

Misal untuk menemukan manajer yang memiliki setidaknya lima bawahan langsung, tambahkan klausa HAVING ke query di atas sebagai berikut:

```
SELECT
    manager_id,
    first_name,
    last_name,
    COUNT(employee_id) direct_reports
FROM
    employees
WHERE
    manager_id IS NOT NULL
GROUP BY manager_id
HAVING direct_reports >= 5;
```

	manager_id	first_name	last_name	direct_reports
▶	100	Neena	Kochhar	10
	101	Nancy	Greenberg	5
	108	Daniel	Faviet	5
	114	Alexander	Khoo	5