Buat Database baru

1. Membuat Database Baru

TESTTT
tess
tess
tesss

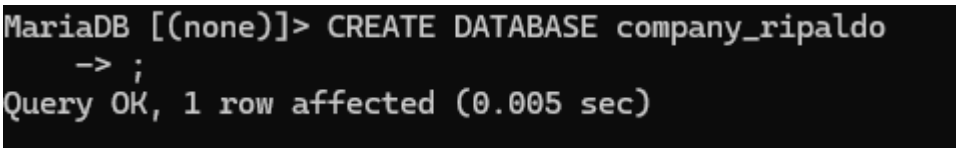
Untuk membuat database baru, gunakan perintah `CREATE DATABASE` di SQL. Berikut adalah contoh sintaks untuk membuat database:

```
CREATE DATABASE nama_database;
```

Contoh :

```
CREATE DATABASE pegawai;
```

Hasil :



- `CREATE DATABASE` : Perintah SQL untuk membuat database baru.
- `nama_database` : Nama yang Anda pilih untuk database baru. Nama ini harus unik dalam server database dan tidak boleh mengandung spasi.

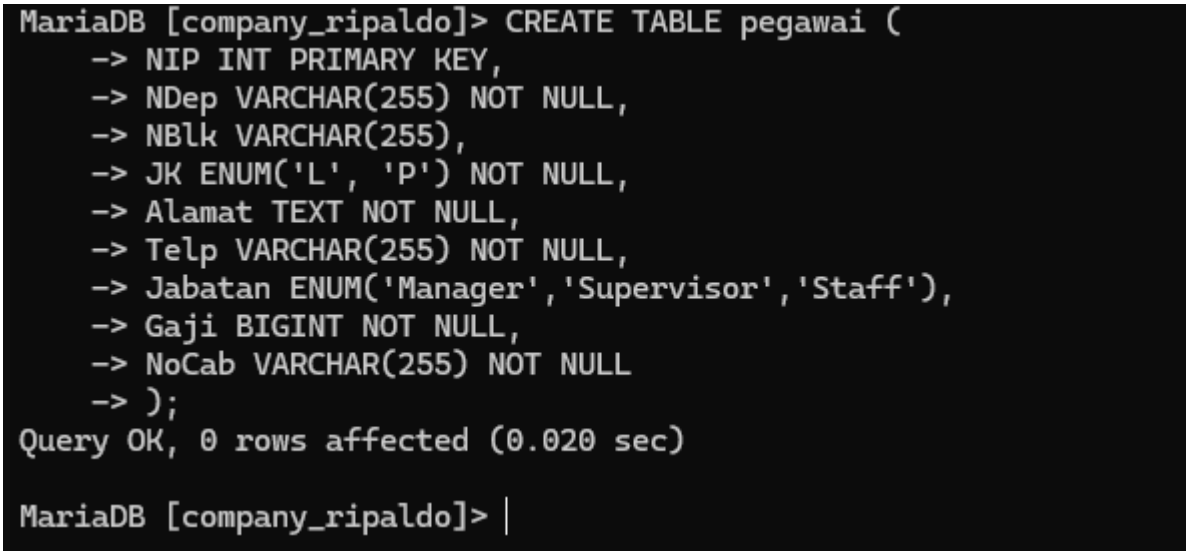
Membuat Table

STRUKTUR "

```
CREATE TABLE pegawai (
-> NIP INT PRIMARY KEY,
-> NDep VARCHAR(255) NOT NULL,
-> NBlk VARCHAR(255),
-> JK ENUM('L', 'P') NOT NULL,
```

```
-> Alamat TEXT NOT NULL,
-> Telp VARCHAR(255) NOT NULL,
Jabatan ENUM('Manager', 'Supervisor', 'Staff'),
    Gaji BIGINT NOT NULL,
    NoCab VARCHAR(255) NOT NULL
);
```

HASIL :



- `NIP` (tipe integer, berfungsi sebagai primary key)
- `NDep` (string yang tidak boleh kosong, mewakili nama depan)
- `NBlk` (string, mewakili nama belakang)
- `JK` (enum dengan nilai 'L' atau 'P', mewakili jenis kelamin, dan tidak boleh kosong)
- `Alamat` (teks yang tidak boleh kosong untuk alamat)
- `Telp` (string yang tidak boleh kosong untuk nomor telepon)
- `Jabatan` (enum dengan nilai 'Manager', 'Supervisor', atau 'Staff')
- `Gaji` (big integer yang tidak boleh kosong, mewakili gaji)
- `NoCab` (string yang tidak boleh kosong, kemungkinan mewakili nomor cabang atau kantor)

Untuk penjelasan lanjutnya kita lanjut di materi selanjutnya. Dengan mengecek Struktur Table....

Mengecek Struktur Table

Untuk mengecek struktur pada table, kita menggunakan perintah `Desc`

Struktur :

```
DESC nama_table
```

Contoh :

```
DESC pegawai
```

```
MariaDB [company_ripaldo]> DESC pegawai;
+---------+-----------------------------------+------+-----+---------+-------+
| Field   | Type                              | Null | Key | Default | Extra |
+---------+-----------------------------------+------+-----+---------+-------+
| NIP     | int(11)                           | NO   | PRI | NULL    |       |
| NDep    | varchar(255)                      | NO   |     | NULL    |       |
| NBlk    | varchar(255)                      | YES  |     | NULL    |       |
| JK      | enum('L','P')                     | NO   |     | NULL    |       |
| Alamat  | text                              | NO   |     | NULL    |       |
| Telp    | varchar(255)                      | NO   |     | NULL    |       |
| Jabatan | enum('Manager','Supervisor','Staff') | YES  |     | NULL    |       |
| Gaji    | bigint(20)                        | NO   |     | NULL    |       |
| NoCab   | varchar(255)                      | NO   |     | NULL    |       |
+---------+-----------------------------------+------+-----+---------+-------+
9 rows in set (0.071 sec)
```

PENJELASAN :

> Deskripsi Tabel Pegawai

1. NIP (INT, PRIMARY KEY)

- NIP: Menyimpan Nomor Induk Pegawai. Tipe data ini adalah integer (INT).
- PRIMARY KEY: Menandakan bahwa NIP adalah kunci utama untuk tabel ini. Artinya, setiap Nomor Induk Pegawai harus unik dan tidak boleh kosong (NULL).

2. NDep (VARCHAR(255), NOT NULL)

- NDep: Menyimpan Nama Depan pegawai. Tipe data ini adalah string dengan panjang maksimum 255 karakter (VARCHAR(255)).
- NOT NULL: Kolom ini harus diisi; tidak boleh kosong (NULL).

3. NBlk (VARCHAR(255))

- NBlk: Menyimpan Nama Belakang pegawai. Tipe data ini adalah string dengan panjang maksimum 255 karakter (VARCHAR(255)).
- NULL YES: Kolom ini bisa kosong (NULL) jika tidak ada nilai.

4. JK (ENUM('L', 'P'), NOT NULL)

- JK: Menyimpan Jenis Kelamin pegawai. Tipe data ini adalah ENUM dengan dua pilihan: 'L' untuk Laki-laki dan 'P' untuk Perempuan.
- NOT NULL: Kolom ini harus diisi dengan salah satu dari nilai yang diperbolehkan; tidak boleh kosong (NULL).

5. Alamat (TEXT, NOT NULL)

- Alamat: Menyimpan alamat pegawai. Tipe data ini adalah TEXT, yang memungkinkan penyimpanan teks dalam jumlah besar.
- NOT NULL: Kolom ini harus diisi; tidak boleh kosong (NULL).

6. Telp (VARCHAR(255), NOT NULL)

- Telp: Menyimpan nomor telepon pegawai. Tipe data ini adalah string dengan panjang maksimum 255 karakter (VARCHAR(255)).
- NOT NULL: Kolom ini harus diisi; tidak boleh kosong (NULL).

7. Jabatan (ENUM('Manajer', 'Supervisor', 'Staf'))

- Jabatan: Menyimpan jabatan pegawai. Tipe data ini adalah ENUM dengan tiga pilihan: 'Manajer', 'Supervisor', atau 'Staf'.
- NULL YES: Kolom ini bisa kosong (NULL) jika tidak ada nilai.

8. Gaji (BIGINT, NOT NULL)

- Gaji: Menyimpan gaji pegawai. Tipe data ini adalah BIGINT, yang memungkinkan penyimpanan angka dalam jumlah sangat besar.
- NOT NULL: Kolom ini harus diisi; tidak boleh kosong (NULL).

9. NoCab (VARCHAR(255), NOT NULL)

- NoCab: Menyimpan nomor cabang pegawai. Tipe data ini adalah string dengan panjang maksimum 255 karakter (VARCHAR(255)).
- NOT NULL: Kolom ini harus diisi; tidak boleh kosong (NULL).

Mengisi data pada table

STRUKTUR "

```
INSERT INTO pegawai (NIP, NDep, NBlk, JK, Alamat, Telp, Jabatan, Gaji, NoCab)
VALUES
-> (10107, 'Emya', 'Salsalina', 'P', 'JL. Suci 78 Bandung', '022-555768',
```

```
'Manager', 5250000, 'C101'),
-> (10246, 'Dian', 'Anggraini', 'P', 'JL. Mawar 5 Semarang', '024-555102',
'Supervisor', 2750000, 'C103'),
-> (10324, 'Martin', 'Susanto', 'L', 'JL. Bima 51 Jakarta', '021-555785', 'Staff',
1750000, 'C102'),
-> (10252, 'Antoni', 'Irawan', 'L', 'JL. A. Yani 15 Jakarta', '021-555888',
'Manager', 5750000, 'C102'),
-> (10176, 'Diah', 'Wahyuni', 'P', 'JL. Maluku 56 Bandung', '022-555934',
'Supervisor', 2500000, 'C101'),
-> (10314, 'Ayu', 'Rahmadani', 'P', 'JL. Malaka 342 Jakarta', '021-555098',
'Supervisor', 1950000, 'C102'),
-> (10307, 'Erik', 'Adrian', 'L', 'JL. Manggis 5 Semarang', '024-555236',
'Manager', 6250000, 'C103'),
-> (10415, 'Susan', 'Sumantri', 'P', 'JL. Pahlawan 24 Surabaya', '031-555120', '',
2650000, 'C104'),
-> (10407, 'Rio', 'Gunawan', 'L', 'JL. Melati 356 Surabaya', '031-555231', 'Staff',
1725000, 'C104');
```

PENJELASAN :

1. `INSERT INTO pegawai` :
   - Ini adalah perintah SQL yang digunakan untuk menambahkan data baru ke tabel yang bernama `pegawai`.

- **Kolom yang Diisi**:
  - Dalam perintah ini, Anda menentukan kolom-kolom di tabel `pegawai` yang akan diisi dengan data. Kolom-kolom tersebut adalah:
    - **NIP**: Nomor Induk Pegawai
    - **NDep**: Nama Depan
    - **NBlk**: Nama Belakang
    - **JK**: Jenis Kelamin
    - **Alamat**: Alamat
    - **Telp**: Nomor Telepon
    - **Jabatan**: Jabatan
    - **Gaji**: Gaji
    - **NoCab**: Nomor Cabang

2. `VALUES` :
   - Setelah menentukan kolom-kolom yang akan diisi, Anda menyebutkan data yang akan dimasukkan. Data untuk setiap baris harus mengikuti urutan kolom yang telah disebutkan.

- **Contoh Data yang Dimasukkan**:
  - **Baris Pertama**:
    - **NIP**: 10107
    - **NDep**: 'Emya'
    - **NBlk**: 'Salsalina'
    - **JK**: 'P' (Perempuan)
    - **Alamat**: 'JL. Suci 78 Bandung'
    - **Telp**: '022-555768'
    - **Jabatan**: 'Manager'
    - **Gaji**: 5.250.000
    - **NoCab**: 'C101'
  - **Baris Berikutnya**:
    - Mengikuti format yang sama dengan data yang berbeda untuk setiap kolom.
- **Catatan Penting**:

```
- **Kolom Jabatan**: Pada baris data dengan **NIP** 10415 untuk **Susan Sumantri**, kolom Jabatan tidak diisi (`''`). Jika kolom Jabatan adalah ENUM (jenis kelamin) dan tidak mengizinkan nilai kosong, ini dapat menyebabkan masalah. Pastikan kolom Jabatan diisi dengan nilai yang valid seperti 'Staff' jika kolom tersebut tidak mengizinkan nilai kosong.
- **Pembaruan Data**: Jika kolom Jabatan mengizinkan nilai kosong, Anda mungkin perlu memperbarui baris ini dengan jabatan yang sesuai. Jika tidak, Anda mungkin perlu menyesuaikan skema tabel untuk mengizinkan nilai kosong.
```

Hasil :

```
MariaDB [company_ripaldo]> INSERT INTO pegawai (NIP,NDep,NBlk,JK,Alamat,Telp,Jabatan,Gaji,NoCab) VALUES
    -> (10107,'Emya','Salsalina','P','Jl.Suci 78 Bandung','022-555768','Manager',5250000,'C101'),
    -> (10246,'Dian','Anggraini','P','Jl.Mawar 5 Semarang','024-555102','Supersivor',2750000,'C103'),
    -> (10324,'Martin','Susanto','L','Jl.Bima 51 Jakarta','021-555785','Staff',1750000,'C102'),
    -> (10252,'Antoni','Irawan','L','Jl.A.Yani 15 Jakarta','021-555888','Manager',5750000,'C102'),
    -> (10176,'Diah','Wahyuni','P','Jl.Maluku 56 Bandung','022-555934','Supervisor',2500000,'C101'),
    -> (10314,'Ayu','Rahmadani','P','Jl.Malaka 324 Jakarta','021-555098','Supervisor',1950000,'C102'),
    -> (10307,'Erik','Adrian','L','Jl.Manggis 5 Semarang','024-555236','Manager',6250000,'C103'),
    -> (10415,'Susan','Sumantri','P','Jl.Pahlawan 24 Surabaya','031-555120','',2650000,'C104'),
    -> (10407,'Rio','Gunawan','L','Jl.Melati 356 Surabaya','031-555231','Staff',1725000,'C104');
Query OK, 9 rows affected, 2 warnings (0.009 sec)
Records: 9  Duplicates: 0  Warnings: 2
```

Menghitung jumlah entri atau baris dalam kolom tertentu

```
MariaDB [company_ripaldo]> SELECT COUNT(NIP) AS JumlahPegawai, COUNT(Jabatan) AS JumlahJabatan FROM pegawai;
+---------------+---------------+
| JumlahPegawai | JumlahJabatan |
+---------------+---------------+
|             9 |             9 |
+---------------+---------------+
1 row in set (0.207 sec)
```

1. SELECT

- Perintah `SELECT` digunakan untuk memilih data dari satu atau lebih tabel dalam database.

2. COUNT(NIP) AS JumlahPegawai

- Fungsi `COUNT()` digunakan untuk menghitung jumlah baris yang berisi nilai non-null dalam kolom yang ditentukan.
- `NIP` adalah nama kolom dalam tabel `pegawai`. Jika tidak ada nilai null dalam kolom ini, fungsi ini akan menghitung jumlah total entri atau baris.
- `AS JumlahPegawai` memberi alias pada hasil dari `COUNT(NIP)`. Ini berarti hasil perhitungan akan diberi nama `JumlahPegawai`, yang dapat digunakan sebagai nama kolom dalam hasil output.

Contoh: Jika ada 100 baris dalam kolom `NIP`, maka hasil dari `COUNT(NIP)` akan menjadi 100, dan ini akan ditampilkan sebagai `JumlahPegawai`.

3. COUNT(Jabatan) AS JumlahJabatan

- Sama seperti pada `COUNT(NIP)`, fungsi `COUNT(Jabatan)` menghitung jumlah entri non-null dalam kolom `Jabatan`.
- `AS JumlahJabatan` memberi alias pada hasil perhitungan sehingga hasil dari `COUNT(Jabatan)` akan ditampilkan dengan nama `JumlahJabatan`.

Contoh: Jika ada 95 baris dalam kolom `Jabatan` (dengan asumsi ada beberapa baris null), maka hasil dari `COUNT(Jabatan)` akan menjadi 95, dan ini akan ditampilkan sebagai `JumlahJabatan`.

4. FROM pegawai

- Perintah ini menentukan tabel tempat pengambilan data. Dalam hal ini, tabel yang dimaksud adalah `pegawai`.

Hasilnya akan menunjukkan dua angka: satu untuk jumlah total pegawai (`JumlahPegawai`) dan satu lagi untuk jumlah pegawai dengan jabatan (`JumlahJabatan`).

Menghitung jumlah pegawai yang terdaftar di cabang tertentu

```
MariaDB [company_ripaldo]> SELECT COUNT(NIP) AS JumlahPegawai
    -> FROM pegawai
    -> WHERE NoCab = 'C102';
+---------------+
| JumlahPegawai |
+---------------+
|             3 |
+---------------+
1 row in set (0.036 sec)
```

- **COUNT(NIP)**: Fungsi ini digunakan untuk menghitung jumlah baris dalam kolom `NIP` yang memiliki nilai non-null. Ini berarti fungsi akan menghitung berapa kali NIP (Nomor Induk Pegawai) muncul dalam tabel `pegawai` . Jika ada NIP yang null, baris tersebut tidak akan dihitung.
- **AS JumlahPegawai**: Bagian ini memberikan alias pada hasil dari `COUNT(NIP)` . Ini berarti hasil perhitungan akan diberi nama `JumlahPegawai` . Jadi, ketika hasilnya ditampilkan, kolom hasil perhitungan akan diberi nama `JumlahPegawai` , yang menggambarkan total pegawai dalam cabang tertentu.
- **FROM pegawai**: Bagian ini menentukan tabel dari mana data akan diambil. Dalam hal ini, data diambil dari tabel `pegawai` , yang merupakan tabel yang menyimpan informasi tentang pegawai.
- **WHERE NoCab = 'C102'**: Bagian ini adalah kondisi yang menyaring data. Hanya baris-baris dalam tabel `pegawai` yang memiliki nilai `NoCab` sama dengan 'C102' yang akan dihitung. `NoCab` adalah kolom yang menyimpan kode cabang, dan 'C102' adalah kode cabang spesifik yang sedang dihitung.

Hasil akhir dari perintah ini akan menampilkan jumlah pegawai yang terdaftar di cabang dengan kode `C102` , dan hasilnya akan ditampilkan dengan nama alias `JumlahPegawai` .

Menghitung jumlah pegawai di setiap cabang yang berbeda

```
MariaDB [company_ripaldo]> SELECT NoCab, COUNT(NIP) AS Jumlah_Pegawai
    -> FROM pegawai
    -> GROUP BY NoCab;
+-------+---------------+
| NoCab | Jumlah_Pegawai |
+-------+---------------+
| C101  |             2 |
| C102  |             3 |
| C103  |             2 |
| C104  |             2 |
+-------+---------------+
4 rows in set (0.029 sec)
```

- **NoCab**: Kolom ini mewakili kode cabang (`NoCab`). Perintah ini akan mengelompokkan data berdasarkan cabang yang berbeda, sehingga hasil akhirnya akan mencantumkan kode cabang bersama dengan jumlah pegawai di cabang tersebut.
- **COUNT(NIP) AS Jumlah_Pegawai**: Fungsi `COUNT(NIP)` menghitung jumlah baris dalam kolom `NIP` yang memiliki nilai non-null, untuk setiap kelompok `NoCab`. Alias `Jumlah_Pegawai` digunakan untuk memberi nama hasil perhitungan ini. Ini berarti hasil perhitungan akan menampilkan jumlah pegawai yang terdaftar di setiap cabang, dan hasilnya akan diberi nama `Jumlah_Pegawai`.
- **FROM pegawai**: Bagian ini menentukan tabel dari mana data diambil, yaitu `pegawai`. Ini adalah tabel yang berisi data tentang pegawai, termasuk informasi tentang cabang mereka (`NoCab`).
- **GROUP BY NoCab**: Bagian ini adalah inti dari perintah, yang mengelompokkan data berdasarkan nilai yang berbeda dalam kolom `NoCab`. Ini berarti, semua baris dengan nilai `NoCab` yang sama akan dikelompokkan bersama, dan kemudian fungsi `COUNT(NIP)` akan diterapkan untuk menghitung jumlah pegawai dalam setiap kelompok tersebut.

Hasil dari perintah ini adalah daftar setiap cabang (`NoCab`) yang ada dalam tabel, bersama dengan jumlah pegawai (`Jumlah_Pegawai`) di setiap cabang tersebut. Setiap baris dalam hasil akan menunjukkan satu cabang dan jumlah pegawai yang bekerja di cabang tersebut.

Menghitung jumlah pegawai di setiap cabang

```
MariaDB [company_ripaldo]> SELECT NoCab, COUNT(NIP) AS Jumlah_Pegawai
    -> FROM pegawai
    -> GROUP BY NoCab HAVING COUNT(NIP) >= 3;
+-------+---------------+
| NoCab | Jumlah_Pegawai |
+-------+---------------+
| C102  |             3 |
+-------+---------------+
1 row in set (0.012 sec)
```

- **NoCab**: Ini merujuk pada kolom dalam tabel yang berisi kode cabang (`NoCab`). Perintah ini akan mengelompokkan hasil berdasarkan cabang yang berbeda.
- **COUNT(NIP) AS Jumlah_pegawai**: Fungsi `COUNT(NIP)` menghitung jumlah baris dalam kolom `NIP` yang memiliki nilai non-null untuk setiap kelompok `NoCab`. Alias `Jumlah_pegawai` digunakan untuk memberi nama hasil perhitungan ini. Dengan demikian, hasilnya akan menampilkan jumlah pegawai di setiap cabang.
- **FROM pegawai**: Ini menunjukkan bahwa data diambil dari tabel `pegawai`, yang merupakan tabel yang berisi informasi tentang pegawai, termasuk cabang mereka (`NoCab`).
- **GROUP BY NoCab**: Bagian ini mengelompokkan data berdasarkan nilai `NoCab`. Semua baris dengan nilai `NoCab` yang sama akan dikelompokkan bersama. Setelah pengelompokan, fungsi `COUNT(NIP)` akan menghitung jumlah pegawai dalam setiap kelompok.
- **HAVING COUNT(NIP) >= 3**: Bagian ini menyaring hasil setelah pengelompokan. Hanya kelompok yang memiliki `COUNT(NIP)` (jumlah pegawai) lebih besar dari atau sama dengan 3 yang akan ditampilkan. Artinya, hanya cabang-cabang yang memiliki minimal 3 pegawai yang akan dimasukkan dalam hasil akhir.

Hasil dari perintah ini adalah daftar cabang (`NoCab`) yang memiliki setidaknya 3 pegawai, bersama dengan jumlah pegawai di masing-masing cabang tersebut (`Jumlah_pegawai`). Cabang yang memiliki kurang dari 3 pegawai tidak akan muncul dalam hasil.

Menghitung total keseluruhan gaji dari semua pegawai

```
MariaDB [company_ripaldo]> SELECT SUM(Gaji) AS Total_Gaji
    -> FROM pegawai;
+------------+
| Total_Gaji |
+------------+
|   30575000 |
+------------+
1 row in set (0.008 sec)
```

- **SUM(Gaji)**: Fungsi `SUM` menjumlahkan semua nilai dalam kolom `Gaji` yang terdapat dalam tabel `pegawai`. Ini berarti total gaji dari semua pegawai yang tercatat dalam tabel akan dihitung.
- **AS Total_Gaji**: Alias `Total_Gaji` digunakan untuk memberi nama hasil dari fungsi `SUM(Gaji)`. Ini berarti hasil dari perhitungan jumlah keseluruhan gaji akan ditampilkan dengan nama `Total_Gaji`.
- **FROM pegawai**: Bagian ini menunjukkan bahwa data yang digunakan untuk perhitungan diambil dari tabel `pegawai`. Tabel ini berisi informasi tentang pegawai, termasuk kolom `Gaji` yang berisi data gaji masing-masing pegawai.

Hasil dari perintah ini adalah satu nilai yang mewakili total keseluruhan gaji dari semua pegawai yang tercatat dalam tabel `pegawai`, dan hasil ini akan ditampilkan dengan nama `Total_Gaji`.

Menghitung total gaji dari semua pegawai yang memiliki jabatan sebagai "Manager"

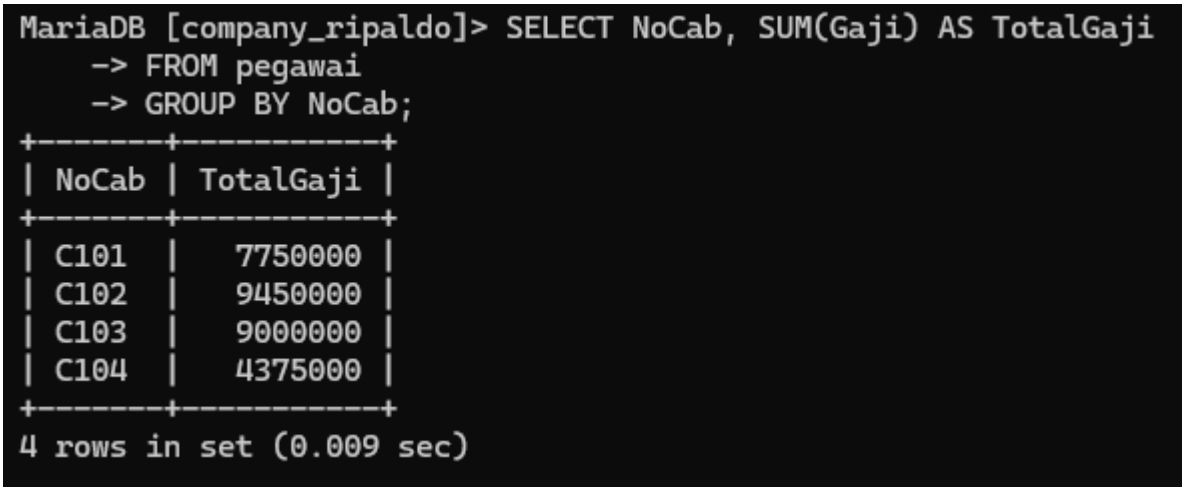
```
MariaDB [company_ripaldo]> SELECT SUM(Gaji) AS Gaji_Manager
    -> FROM pegawai
    -> WHERE Jabatan = 'Manager';
+--------------+
| Gaji_Manager |
+--------------+
|     17250000 |
+--------------+
1 row in set (0.004 sec)
```

- **SUM(Gaji)**: Fungsi `SUM` digunakan untuk menjumlahkan semua nilai dalam kolom `Gaji` yang memenuhi kondisi tertentu, yaitu yang memiliki jabatan "Manager". Ini berarti total gaji dari semua pegawai dengan jabatan "Manager" akan dihitung.
- **AS Gaji_Manager**: Alias `Gaji_Manager` digunakan untuk memberi nama hasil dari fungsi `SUM(Gaji)`. Hasil ini akan menampilkan total gaji dari semua "Manager" dalam tabel, dengan nama alias `Gaji_Manager`.

- **FROM pegawai**: Bagian ini menunjukkan bahwa data yang digunakan diambil dari tabel `pegawai`, yang berisi informasi tentang pegawai, termasuk jabatan dan gaji mereka.
- **WHERE Jabatan = 'Manager'**: Kondisi `WHERE` ini menyaring data sehingga hanya baris-baris yang memiliki nilai `Jabatan` yang sama dengan "Manager" yang akan dipertimbangkan dalam perhitungan. Hanya pegawai dengan jabatan "Manager" yang gajinya akan dijumlahkan.

Hasil dari perintah ini adalah satu nilai yang mewakili total keseluruhan gaji dari semua pegawai dengan jabatan "Manager" yang tercatat dalam tabel `pegawai`, dan hasil ini akan ditampilkan dengan nama `Gaji_Manager`.

Menghitung total gaji yang dikeluarkan untuk setiap cabang



```
MariaDB [company_ripaldo]> SELECT NoCab, SUM(Gaji) AS TotalGaji
    -> FROM pegawai
    -> GROUP BY NoCab;
+-------+-----------+
| NoCab | TotalGaji |
+-------+-----------+
| C101  |   7750000 |
| C102  |   9450000 |
| C103  |   9000000 |
| C104  |   4375000 |
+-------+-----------+
4 rows in set (0.009 sec)
```

- **NoCab**: Kolom ini mewakili kode cabang dalam tabel `pegawai`. Perintah ini akan mengelompokkan data berdasarkan nilai dalam kolom `NoCab`.
- **SUM(Gaji) AS TotalGaji**: Fungsi `SUM(Gaji)` menghitung jumlah total nilai dalam kolom `Gaji` untuk setiap kelompok yang dikelompokkan berdasarkan `NoCab`. Alias `TotalGaji` digunakan untuk memberi nama hasil perhitungan ini, sehingga hasilnya akan menampilkan total gaji untuk setiap cabang dengan nama alias `TotalGaji`.
- **FROM pegawai**: Bagian ini menentukan tabel dari mana data diambil. Dalam hal ini, data diambil dari tabel `pegawai`, yang berisi informasi tentang pegawai termasuk cabang dan gaji mereka.
- **GROUP BY NoCab**: Bagian ini mengelompokkan hasil berdasarkan kolom `NoCab`. Semua baris dengan nilai `NoCab` yang sama akan digabungkan bersama, dan kemudian fungsi `SUM(Gaji)` akan menghitung total gaji untuk setiap kelompok cabang.

Hasil dari perintah ini adalah daftar setiap cabang (`NoCab`) yang ada dalam tabel, bersama dengan total gaji (`TotalGaji`) yang dikeluarkan untuk pegawai di masing-masing cabang tersebut. Setiap baris dalam hasil akan menunjukkan satu cabang dan jumlah total gaji yang dibayarkan di cabang tersebut.

Menghitung total gaji di setiap cabang dan hanya menampilkan cabang-cabang yang memiliki total gaji tertentu

```
MariaDB [company_ripaldo]> SELECT NoCab, SUM(Gaji) AS Total_Gaji
    -> FROM pegawai
    -> GROUP BY NoCab HAVING SUM(Gaji) >= 8000000;
+-------+------------+
| NoCab | Total_Gaji |
+-------+------------+
| C102  |    9450000 |
| C103  |    9000000 |
+-------+------------+
2 rows in set (0.008 sec)
```

- **NoCab**: Kolom ini mewakili kode cabang dalam tabel `pegawai` . Perintah ini akan mengelompokkan hasil berdasarkan kode cabang.
- **SUM(Gaji) AS Total_Gaji**: Fungsi `SUM(Gaji)` menghitung total gaji untuk setiap kelompok berdasarkan `NoCab` . Alias `Total_Gaji` digunakan untuk memberi nama hasil perhitungan ini, sehingga total gaji untuk setiap cabang akan ditampilkan dengan nama `Total_Gaji` .
- **FROM pegawai**: Bagian ini menunjukkan bahwa data diambil dari tabel `pegawai` , yang menyimpan informasi tentang pegawai, termasuk cabang dan gaji mereka.
- **GROUP BY NoCab**: Bagian ini mengelompokkan data berdasarkan kolom `NoCab` . Semua baris dengan nilai `NoCab` yang sama akan digabungkan, dan fungsi `SUM(Gaji)` akan menghitung total gaji untuk setiap kelompok cabang.
- **HAVING SUM(Gaji) >= 8000000**: Kondisi `HAVING` menyaring hasil setelah pengelompokan. Hanya cabang-cabang yang memiliki total gaji (`SUM(Gaji)`) lebih besar dari atau sama dengan 8.000.000 yang akan ditampilkan dalam hasil.

Hasil dari perintah ini adalah daftar cabang (`NoCab`) yang memiliki total gaji pegawai sebesar atau lebih dari 8.000.000, bersama dengan jumlah total gaji (`Total_Gaji`) yang dibayarkan di masing-masing cabang tersebut. Cabang-cabang yang total gajinya kurang dari 8.000.000 tidak akan muncul dalam hasil.

Menghitung rata-rata gaji dari semua pegawai yang terdaftar

```
MariaDB [company_ripaldo]> SELECT AVG(Gaji) AS Rata_rata
    -> FROM pegawai;
+--------------+
| Rata_rata    |
+--------------+
| 3397222.2222 |
+--------------+
1 row in set (0.001 sec)
```

- **AVG(Gaji)**: Fungsi `AVG` menghitung rata-rata dari nilai-nilai dalam kolom `Gaji`. Ini berarti fungsi ini akan menjumlahkan semua nilai gaji dan membagi hasilnya dengan jumlah baris yang memiliki nilai non-null dalam kolom `Gaji`.
- **AS Rata_rata**: Alias `Rata_rata` digunakan untuk memberi nama pada hasil dari fungsi `AVG(Gaji)`. Ini berarti hasil rata-rata gaji akan ditampilkan dengan nama `Rata_rata`.
- **FROM pegawai**: Bagian ini menunjukkan bahwa data yang digunakan untuk perhitungan diambil dari tabel `pegawai`, yang berisi informasi tentang pegawai, termasuk kolom `Gaji`.

Hasil dari perintah ini adalah satu nilai yang mewakili rata-rata gaji dari semua pegawai yang tercatat dalam tabel `pegawai`, dan hasil ini akan ditampilkan dengan nama `Rata_rata`.

Menghitung rata-rata gaji dari pegawai yang memiliki jabatan tertentu

```
MariaDB [company_ripaldo]> SELECT AVG(Gaji) AS GajiRataMgr
    -> FROM pegawai
    -> WHERE Jabatan = 'Manager';
+--------------+
| GajiRataMgr  |
+--------------+
| 5750000.0000 |
+--------------+
1 row in set (0.001 sec)
```

- **AVG(Gaji)**: Fungsi `AVG` digunakan untuk menghitung rata-rata nilai dalam kolom `Gaji`. Fungsi ini akan menjumlahkan semua nilai gaji untuk pegawai yang memenuhi kondisi yang ditentukan dan kemudian membagi jumlah tersebut dengan jumlah baris yang memenuhi kondisi.
- **AS GajiRatamgr**: Alias `GajiRatamgr` digunakan untuk memberi nama hasil dari fungsi `AVG(Gaji)`. Ini berarti hasil perhitungan rata-rata gaji untuk pegawai dengan jabatan

"Manager" akan ditampilkan dengan nama `GajiRatamgr`.

- **FROM pegawai**: Bagian ini menunjukkan bahwa data yang digunakan diambil dari tabel `pegawai`, yang berisi informasi tentang pegawai termasuk kolom `Gaji` dan `Jabatan`.
- **WHERE Jabatan = 'Manager'**: Kondisi `WHERE` ini menyaring data sehingga hanya baris-baris dengan nilai `Jabatan` yang sama dengan "Manager" yang akan dipertimbangkan dalam perhitungan. Ini berarti hanya gaji dari pegawai yang memiliki jabatan "Manager" yang akan dihitung untuk rata-rata.

Hasil dari perintah ini adalah satu nilai yang mewakili rata-rata gaji dari semua pegawai yang memiliki jabatan "Manager" dalam tabel `pegawai`, dan hasil ini akan ditampilkan dengan nama alias `GajiRatamgr`.

Menghitung rata-rata gaji pegawai di setiap cabang

```
MariaDB [company_ripaldo]> SELECT NoCab, AVG(Gaji) AS RataGaji
    -> FROM pegawai
    -> GROUP BY NoCab;
+-------+--------------+
| NoCab | RataGaji     |
+-------+--------------+
| C101  | 3875000.0000 |
| C102  | 3150000.0000 |
| C103  | 4500000.0000 |
| C104  | 2187500.0000 |
+-------+--------------+
4 rows in set (0.004 sec)
```

- **NoCab**: Kolom ini merujuk pada kode cabang dalam tabel `pegawai`. Perintah ini akan mengelompokkan data berdasarkan nilai dalam kolom `NoCab`.
- **AVG(Gaji) AS RataGaji**: Fungsi `AVG(Gaji)` digunakan untuk menghitung rata-rata nilai dalam kolom `Gaji` untuk setiap kelompok yang dikelompokkan berdasarkan `NoCab`. Alias `RataGaji` digunakan untuk memberi nama pada hasil perhitungan ini, sehingga hasil rata-rata gaji untuk setiap cabang akan ditampilkan dengan nama `RataGaji`.
- **FROM pegawai**: Bagian ini menunjukkan bahwa data yang digunakan diambil dari tabel `pegawai`, yang berisi informasi tentang pegawai, termasuk kolom `Gaji` dan `NoCab`.
- **GROUP BY NoCab**: Bagian ini mengelompokkan hasil berdasarkan kolom `NoCab`. Semua baris dengan nilai `NoCab` yang sama akan digabungkan bersama, dan fungsi `AVG(Gaji)` akan menghitung rata-rata gaji untuk setiap kelompok cabang.

Hasil dari perintah ini adalah daftar setiap cabang (`NoCab`) yang ada dalam tabel, bersama dengan rata-rata gaji (`RataGaji`) yang dibayarkan di masing-masing cabang tersebut. Setiap baris dalam hasil akan menunjukkan satu cabang dan rata-rata gaji pegawai di cabang tersebut.

Menghitung rata-rata gaji pegawai di cabang-cabang tertentu

```
MariaDB [company_ripaldo]> SELECT NoCab, AVG(Gaji) AS RataGaji
    -> FROM pegawai
    -> GROUP BY NoCab HAVING NoCab = 'C101' OR NoCab = 'C102';
+-------+-------------+
| NoCab | RataGaji    |
+-------+-------------+
| C101  | 3875000.0000 |
| C102  | 3150000.0000 |
+-------+-------------+
2 rows in set (0.014 sec)
```

- **NoCab**: Kolom ini merujuk pada kode cabang dalam tabel `pegawai` . Perintah ini akan mengelompokkan data berdasarkan nilai dalam kolom `NoCab` .
- **AVG(Gaji) AS RataGaji**: Fungsi `AVG(Gaji)` digunakan untuk menghitung rata-rata nilai dalam kolom `Gaji` untuk setiap kelompok yang dikelompokkan berdasarkan `NoCab` . Alias `RataGaji` digunakan untuk memberi nama hasil perhitungan ini, sehingga rata-rata gaji untuk setiap cabang akan ditampilkan dengan nama `RataGaji` .
- **FROM pegawai**: Bagian ini menunjukkan bahwa data yang digunakan diambil dari tabel `pegawai` , yang berisi informasi tentang pegawai, termasuk kolom `Gaji` dan `NoCab` .
- **GROUP BY NoCab**: Bagian ini mengelompokkan hasil berdasarkan kolom `NoCab` . Semua baris dengan nilai `NoCab` yang sama akan digabungkan bersama, dan fungsi `AVG(Gaji)` akan menghitung rata-rata gaji untuk setiap kelompok cabang.
- **HAVING NoCab = 'C101' OR NoCab = 'C102'**: Kondisi `HAVING` menyaring hasil setelah pengelompokan. Hanya cabang-cabang dengan kode `NoCab` yang sama dengan 'C101' atau 'C102' yang akan ditampilkan dalam hasil. Baris-baris dengan kode cabang lainnya tidak akan muncul dalam hasil.

Hasil dari perintah ini adalah daftar cabang-cabang yang memiliki kode 'C101' atau 'C102', bersama dengan rata-rata gaji (`RataGaji`) dari pegawai di cabang-cabang tersebut.

Menemukan nilai gaji terbesar dan terkecil dari semua pegawai

```
MariaDB [company_ripaldo]> SELECT MAX(Gaji) AS GajiTerbesar, MIN(Gaji) AS GajiTerkecil
    -> FROM pegawai;
+--------------+--------------+
| GajiTerbesar | GajiTerkecil |
+--------------+--------------+
|      6250000 |      1725000 |
+--------------+--------------+
1 row in set (0.007 sec)
```

- **MAX(Gaji) AS GajiTerbesar**: Fungsi `MAX` digunakan untuk menemukan nilai maksimum dalam kolom `Gaji`. Ini berarti fungsi ini akan mengidentifikasi gaji tertinggi yang ada dalam tabel `pegawai`. Alias `GajiTerbesar` digunakan untuk memberi nama pada hasil dari fungsi `MAX(Gaji)`, sehingga hasilnya akan ditampilkan dengan nama `GajiTerbesar`.
- **MIN(Gaji) AS GajiTerkecil**: Fungsi `MIN` digunakan untuk menemukan nilai minimum dalam kolom `Gaji`. Ini berarti fungsi ini akan mengidentifikasi gaji terendah yang ada dalam tabel `pegawai`. Alias `GajiTerkecil` digunakan untuk memberi nama pada hasil dari fungsi `MIN(Gaji)`, sehingga hasilnya akan ditampilkan dengan nama `GajiTerkecil`.
- **FROM pegawai**: Bagian ini menunjukkan bahwa data yang digunakan diambil dari tabel `pegawai`, yang berisi informasi tentang pegawai, termasuk kolom `Gaji`.

Hasil dari perintah ini adalah dua nilai:

- **GajiTerbesar**: Nilai ini menunjukkan gaji tertinggi di antara semua pegawai yang terdaftar dalam tabel `pegawai`.
- **GajiTerkecil**: Nilai ini menunjukkan gaji terendah di antara semua pegawai yang terdaftar dalam tabel `pegawai`.

Kedua nilai ini memberikan gambaran mengenai rentang gaji dalam tabel tersebut.

Menemukan nilai gaji terbesar dan terkecil dari pegawai yang memiliki jabatan "Manager"

```
MariaDB [company_ripaldo]> SELECT MAX(Gaji) AS GajiTerbesar, MIN(Gaji) AS GajiTerkecil
    -> FROM pegawai
    -> WHERE Jabatan = 'Manager';
+--------------+--------------+
| GajiTerbesar | GajiTerkecil |
+--------------+--------------+
|      6250000 |      5250000 |
+--------------+--------------+
1 row in set (0.001 sec)
```

- **MAX(Gaji) AS GajiTerbesar**: Fungsi `MAX` digunakan untuk menemukan nilai maksimum dalam kolom `Gaji` untuk pegawai yang memenuhi kondisi. Ini berarti fungsi ini akan mengidentifikasi gaji tertinggi di antara pegawai yang memiliki jabatan "Manager". Alias `GajiTerbesar` digunakan untuk memberi nama pada hasil dari fungsi `MAX(Gaji)`, sehingga hasilnya akan ditampilkan dengan nama `GajiTerbesar`.
- **MIN(Gaji) AS GajiTerkecil**: Fungsi `MIN` digunakan untuk menemukan nilai minimum dalam kolom `Gaji` untuk pegawai yang memenuhi kondisi. Ini berarti fungsi ini akan mengidentifikasi gaji terendah di antara pegawai yang memiliki jabatan "Manager". Alias `GajiTerkecil` digunakan untuk memberi nama pada hasil dari fungsi `MIN(Gaji)`, sehingga hasilnya akan ditampilkan dengan nama `GajiTerkecil`.
- **FROM pegawai**: Bagian ini menunjukkan bahwa data yang digunakan diambil dari tabel `pegawai`, yang berisi informasi tentang pegawai, termasuk kolom `Gaji` dan `Jabatan`.
- **WHERE Jabatan = 'Manager'**: Kondisi `WHERE` ini menyaring data sehingga hanya baris-baris dengan nilai `Jabatan` yang sama dengan 'Manager' yang akan dipertimbangkan dalam perhitungan. Hanya gaji dari pegawai yang memiliki jabatan "Manager" yang akan dihitung untuk nilai maksimum dan minimum.

Hasil dari perintah ini adalah dua nilai:

- **GajiTerbesar**: Nilai ini menunjukkan gaji tertinggi di antara pegawai yang memiliki jabatan "Manager" yang berjumlah "6250000"
- **GajiTerkecil**: Nilai ini menunjukkan gaji terendah di antara pegawai yang memiliki jabatan "Manager" yang berjumlah "5250000"

Kedua nilai ini memberikan gambaran mengenai rentang gaji untuk pegawai yang memiliki jabatan "Manager" dalam tabel `pegawai`.

Menemukan gaji terbesar dan terkecil untuk setiap cabang yang memiliki tiga pegawai atau lebih

```
MariaDB [company_ripaldo]> SELECT NoCab, MAX(Gaji) AS GajiTerbesar, MIN(Gaji) AS GajiTerkecil
    -> FROM pegawai
    -> GROUP BY NoCab;
+-------+--------------+--------------+
| NoCab | GajiTerbesar | GajiTerkecil |
+-------+--------------+--------------+
| C101  |      5250000 |      2500000 |
| C102  |      5750000 |      1750000 |
| C103  |      6250000 |      2750000 |
| C104  |      2650000 |      1725000 |
+-------+--------------+--------------+
4 rows in set (0.011 sec)
```

- **NoCab**: Kolom ini merujuk pada kode cabang dalam tabel `pegawai`. Perintah ini akan mengelompokkan data berdasarkan nilai dalam kolom `NoCab`.
- **MAX(Gaji) AS GajiTerbesar**: Fungsi `MAX` digunakan untuk menemukan nilai maksimum dalam kolom `Gaji` untuk setiap kelompok cabang. Alias `GajiTerbesar` digunakan untuk memberi nama pada hasil dari fungsi `MAX(Gaji)`, sehingga hasilnya akan ditampilkan dengan nama `GajiTerbesar`.
- **MIN(Gaji) AS GajiTerkecil**: Fungsi `MIN` digunakan untuk menemukan nilai minimum dalam kolom `Gaji` untuk setiap kelompok cabang. Alias `GajiTerkecil` digunakan untuk memberi nama pada hasil dari fungsi `MIN(Gaji)`, sehingga hasilnya akan ditampilkan dengan nama `GajiTerkecil`.
- **FROM pegawai**: Bagian ini menunjukkan bahwa data yang digunakan diambil dari tabel `pegawai`, yang berisi informasi tentang pegawai, termasuk kolom `Gaji` dan `NoCab`.
- **GROUP BY NoCab**: Bagian ini mengelompokkan hasil berdasarkan kolom `NoCab`. Semua baris dengan nilai `NoCab` yang sama akan digabungkan bersama, dan fungsi `MAX(Gaji)` serta `MIN(Gaji)` akan dihitung untuk setiap kelompok cabang.
- **HAVING COUNT(NIP) >= 3**: Kondisi `HAVING` menyaring hasil setelah pengelompokan. Hanya cabang-cabang yang memiliki tiga pegawai atau lebih (dalam hal ini, dihitung dengan `COUNT(NIP)`) yang akan ditampilkan dalam hasil. Baris-baris dengan jumlah pegawai kurang dari tiga tidak akan muncul dalam hasil.

Hasil dari perintah ini adalah daftar setiap cabang (`NoCab`) yang memiliki tiga pegawai atau lebih, bersama dengan gaji tertinggi (`GajiTerbesar`) dan gaji terendah (`GajiTerkecil`) dari pegawai di setiap cabang tersebut.

Mendapatkan berbagai statistik terkait gaji dari seluruh pegawai

```
MariaDB [company_ripaldo]> SELECT NoCab, MAX(Gaji) GajiTerbesar, MIN(Gaji) GajiTerkecil
    -> FROM pegawai
    -> GROUP BY NoCab HAVING COUNT(NIP) >= 3;
+-------+-------------+-------------+
| NoCab | GajiTerbesar | GajiTerkecil |
+-------+-------------+-------------+
| C102  |      5750000 |      1750000 |
+-------+-------------+-------------+
1 row in set (0.001 sec)
```

- **COUNT(NIP) AS JumlahPegawai**: Fungsi `COUNT` digunakan untuk menghitung jumlah pegawai berdasarkan kolom `NIP`. Alias `JumlahPegawai` digunakan untuk memberi nama pada hasil perhitungan ini. Ini memberikan jumlah total pegawai yang terdaftar dalam tabel `pegawai`.
- **SUM(Gaji) AS TotalGaji**: Fungsi `SUM` digunakan untuk menghitung jumlah total dari kolom `Gaji`. Alias `TotalGaji` digunakan untuk memberi nama pada hasil perhitungan ini. Ini memberikan total akumulasi gaji yang dibayarkan kepada semua pegawai.
- **AVG(Gaji) AS RataGaji**: Fungsi `AVG` digunakan untuk menghitung rata-rata nilai dalam kolom `Gaji`. Alias `RataGaji` digunakan untuk memberi nama pada hasil perhitungan ini. Ini memberikan rata-rata gaji pegawai di seluruh tabel.
- **MAX(Gaji) AS GajiMaks**: Fungsi `MAX` digunakan untuk menemukan nilai maksimum dalam kolom `Gaji`. Alias `GajiMaks` digunakan untuk memberi nama pada hasil perhitungan ini. Ini memberikan gaji tertinggi yang diterima oleh pegawai dalam tabel.
- **MIN(Gaji) AS GajiMin**: Fungsi `MIN` digunakan untuk menemukan nilai minimum dalam kolom `Gaji`. Alias `GajiMin` digunakan untuk memberi nama pada hasil perhitungan ini. Ini memberikan gaji terendah yang diterima oleh pegawai dalam tabel.
- **FROM pegawai**: Bagian ini menunjukkan bahwa data yang digunakan diambil dari tabel `pegawai`, yang berisi informasi tentang pegawai, termasuk kolom `NIP` dan `Gaji`.

Hasil dari perintah ini adalah satu baris data yang menunjukkan:

- **JumlahPegawai**: Total jumlah pegawai yang terdaftar.
- **TotalGaji**: Jumlah total gaji yang dibayarkan kepada semua pegawai.
- **RataGaji**: Rata-rata gaji pegawai.
- **GajiMaks**: Gaji tertinggi yang diterima oleh pegawai.
- **GajiMin**: Gaji terendah yang diterima oleh pegawai.

Perintah ini memberikan gambaran menyeluruh tentang distribusi dan rentang gaji pegawai dalam tabel `pegawai`.

Memperoleh ringkasan statistik terkait data pegawai

```
MariaDB [company_ripaldo]> SELECT COUNT(NIP) AS JumlahPegawai, SUM(Gaji) AS TotalGaji,
    -> AVG(Gaji) AS RataGaji, MAX(Gaji) AS GajiMaks, MIN(Gaji) AS GajiMin
    -> FROM pegawai;
+---------------+-----------+--------------+----------+---------+
| JumlahPegawai | TotalGaji | RataGaji     | GajiMaks | GajiMin |
+---------------+-----------+--------------+----------+---------+
|             9 |  30575000 | 3397222.2222 |  6250000 | 1725000 |
+---------------+-----------+--------------+----------+---------+
1 row in set (0.001 sec)
```

- **COUNT(NIP) AS JumlahPegawai**
  Fungsi `COUNT(NIP)` menghitung jumlah total pegawai berdasarkan kolom `NIP`, yang umumnya adalah identifikasi unik setiap pegawai. Hasil dari fungsi ini memberikan jumlah total pegawai yang ada dalam tabel `pegawai`. Hasil ini diberi alias `JumlahPegawai` untuk memudahkan pembacaan.
- **SUM(Gaji) AS TotalGaji**
  Fungsi `SUM(Gaji)` menjumlahkan semua nilai dalam kolom `Gaji`, memberikan total keseluruhan gaji yang dibayarkan kepada seluruh pegawai. Hasil dari fungsi ini adalah jumlah total gaji yang diterima oleh pegawai di tabel. Hasil ini diberi alias `TotalGaji`.
- **AVG(Gaji) AS RataGaji**
  Fungsi `AVG(Gaji)` menghitung rata-rata nilai dalam kolom `Gaji`, memberikan rata-rata gaji yang diterima oleh pegawai. Perhitungan dilakukan dengan menjumlahkan seluruh nilai gaji dan membaginya dengan jumlah pegawai. Hasil ini diberi alias `RataGaji`.
- **MAX(Gaji) AS GajiMaks**
  Fungsi `MAX(Gaji)` mencari nilai maksimum dari kolom `Gaji`, yaitu gaji tertinggi yang diterima oleh pegawai. Hasil ini diberi alias `GajiMaks` untuk menunjukkan gaji tertinggi di tabel.
- **MIN(Gaji) AS GajiMin**
  Fungsi `MIN(Gaji)` mencari nilai minimum dari kolom `Gaji`, yaitu gaji terendah yang diterima oleh pegawai. Hasil ini diberi alias `GajiMin` untuk menunjukkan gaji terendah di tabel.

Perintah SQL ini menghasilkan satu baris data yang mencakup jumlah total pegawai, total keseluruhan gaji, rata-rata gaji, gaji tertinggi, dan gaji terendah dari tabel `pegawai`.

Menghitung dan menganalisis statistik gaji pegawai

```
MariaDB [company_ripaldo]> SELECT COUNT(NIP) AS JumlahPegawai, SUM(Gaji) AS TotalGaji,
    -> AVG(Gaji) AS RataGaji, MAX(Gaji) AS GajiMaks, MIN(Gaji) GajiMin
    -> FROM pegawai
    -> WHERE Jabatan = 'Staff' OR Jabatan = 'Sales'
    -> GROUP BY NoCab HAVING SUM(Gaji) <= 2600000;
+--------------+-----------+--------------+----------+---------+
| JumlahPegawai | TotalGaji | RataGaji     | GajiMaks | GajiMin |
+--------------+-----------+--------------+----------+---------+
|            1 |   1750000 | 1750000.0000 |  1750000 | 1750000 |
|            1 |   1725000 | 1725000.0000 |  1725000 | 1725000 |
+--------------+-----------+--------------+----------+---------+
2 rows in set (0.007 sec)
```

- **COUNT(NIP) AS JumlahPegawai**
  Fungsi `COUNT(NIP)` menghitung jumlah pegawai yang memiliki nilai pada kolom `NIP`. Ini memberikan jumlah total pegawai yang memenuhi kondisi dalam klausa `WHERE`. Hasil ini diberi alias `JumlahPegawai`.
- **SUM(Gaji) AS TotalGaji**
  Fungsi `SUM(Gaji)` menghitung total keseluruhan gaji dari pegawai yang memenuhi kondisi dalam klausa `WHERE`. Ini memberikan jumlah akumulatif gaji untuk pegawai yang jabatan-nya adalah 'staf' atau 'Sales'. Hasil ini diberi alias `TotalGaji`.
- **AVG(Gaji) AS RataGaji**
  Fungsi `AVG(Gaji)` menghitung rata-rata gaji dari pegawai yang memenuhi kondisi dalam klausa `WHERE`. Ini memberikan nilai rata-rata gaji untuk pegawai dengan jabatan 'staf' atau 'Sales'. Hasil ini diberi alias `RataGaji`.
- **MAX(Gaji) AS GajiMaks**
  Fungsi `MAX(Gaji)` mencari gaji tertinggi dari pegawai yang memenuhi kondisi dalam klausa `WHERE`. Ini memberikan nilai maksimum gaji di antara pegawai dengan jabatan 'staf' atau 'Sales'. Hasil ini diberi alias `GajiMaks`.
- **MIN(Gaji) AS GajiMin**
  Fungsi `MIN(Gaji)` mencari gaji terendah dari pegawai yang memenuhi kondisi dalam klausa `WHERE`. Ini memberikan nilai minimum gaji di antara pegawai dengan jabatan 'staf' atau 'Sales'. Hasil ini diberi alias `GajiMin`.

Klausul `WHERE`:
Klausul `WHERE` membatasi data yang dipilih hanya pada pegawai yang memiliki jabatan 'staf' atau 'Sales'.

Klausul `GROUP BY NoCab`:
Data dikelompokkan berdasarkan kolom `NoCab`, yang kemungkinan adalah kode cabang. Ini berarti statistik dihitung untuk setiap cabang secara terpisah.

Klausul `HAVING SUM(Gaji) <= 2600000`:
Klausul `HAVING` digunakan untuk memfilter hasil agregat yang telah dikelompokkan. Dalam hal

ini, hanya cabang yang memiliki total gaji yang kurang dari atau sama dengan 2.600.000 yang akan ditampilkan.

Hasil dari Perintah Ini:
Perintah ini menghasilkan statistik agregat (jumlah pegawai, total gaji, rata-rata gaji, gaji tertinggi, dan gaji terendah) untuk setiap cabang di mana jabatan pegawai adalah 'staf' atau 'Sales', dan total gaji di setiap cabang tidak melebihi 2.600.000.