Pembuatan Database Dan Tabel Pegawai

1. Contoh Tabel Yang Harus Dibuat

2. Struktur Tabel Yang Di Buat

```
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', 'Sales', 'Staff'),
-> Gaji BIGINT NOT NULL,
-> NoCab VARCHAR(255) NOT NULL
);
```

Penjelasan :

- *NIP INT PRIMARY KEY*:
  - *NIP*: Kolom ini menyimpan Nomor Induk Pegawai yang bertipe data integer (INT).
  - *PRIMARY KEY*: Menandakan bahwa kolom NIP adalah kunci utama dari tabel ini. Kunci utama harus memiliki nilai yang unik untuk setiap baris dan tidak boleh NULL.
- *NDep VARCHAR(255) NOT NULL*:
  - *NDep*: Kolom ini menyimpan Nama Depan pegawai bertipe data string dengan panjang maksimum 255 karakter (VARCHAR(255)).
  - *NOT NULL*: Menandakan bahwa kolom ini tidak boleh berisi nilai kosong (NULL); artinya, setiap baris harus memiliki nilai untuk kolom ini.
- *NBlk VARCHAR(255)*:
  - *NBlk*: Kolom ini menyimpan Nama Belakang pegawai bertipe data string dengan panjang maksimum 255 karakter (VARCHAR(255)).
  - Kolom ini tidak memiliki batasan NOT NULL, sehingga nilai kosong (NULL) diizinkan.
- *JK ENUM('L', 'P') NOT NULL*:
  - *JK*: Kolom ini menyimpan Jenis Kelamin pegawai dengan tipe data ENUM. Hanya dua nilai yang diperbolehkan: 'L' (Laki-laki) atau 'P' (Perempuan).
  - *NOT NULL*: Menandakan bahwa kolom ini tidak boleh kosong; artinya, setiap baris harus memiliki nilai untuk kolom ini.
- *Alamat TEXT NOT NULL*:

    - *Alamat*: Kolom ini menyimpan alamat pegawai dengan tipe data TEXT, yang bisa menampung teks dalam jumlah besar.
    - *NOT NULL*: Menandakan bahwa kolom ini harus diisi dengan nilai; nilai kosong (NULL) tidak diperbolehkan.
- *Telp VARCHAR(255) NOT NULL*:
    - *Telp*: Kolom ini menyimpan nomor telepon pegawai dengan tipe data string dan panjang maksimum 255 karakter (VARCHAR(255)).
    - *NOT NULL*: Menandakan bahwa kolom ini harus diisi; nilai kosong (NULL) tidak diperbolehkan.
- *Jabatan ENUM('Manager', 'Sales', 'Staff')*:
    - *Jabatan*: Kolom ini menyimpan jabatan pegawai dengan tipe data ENUM. Nilai yang diperbolehkan adalah 'Manager', 'Sales', atau 'Staff'.
    - Kolom ini tidak memiliki batasan NOT NULL, sehingga nilai kosong (NULL) diizinkan.
- *Gaji BIGINT NOT NULL*:
    - *Gaji*: Kolom ini menyimpan gaji pegawai dengan tipe data BIGINT, yang dapat menampung bilangan bulat yang sangat besar.
    - *NOT NULL*: Menandakan bahwa kolom ini harus diisi; nilai kosong (NULL) tidak diperbolehkan.
- *NoCab VARCHAR(255) NOT NULL*:
    - *NoCab*: Kolom ini menyimpan nomor cabang pegawai dengan tipe data string dan panjang maksimum 255 karakter (VARCHAR(255)).
    - *NOT NULL*: Menandakan bahwa kolom ini harus diisi; nilai kosong (NULL) tidak diperbolehkan.

Hasil :

“BASDAT/Screenshot (33).png” could not be found.

3. Isian Tabel Yang Telah Di Buat

```
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', 'Sales',
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', 'Sales',
```

```
2500000, 'C101'),
-> (10314, 'Ayu', 'Rahmadani', 'P', 'JL. Malaka 342 Jakarta', '021-555098',
'Sales', 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 :

- *INSERT INTO pegawai*:
  - Menunjukkan bahwa Anda akan menambahkan data ke tabel bernama pegawai.
- *(NIP, NDep, NBlk, JK, Alamat, Telp, Jabatan, Gaji, NoCab)*:
  - Ini adalah daftar kolom dalam tabel pegawai yang akan diisi dengan data. Kolom-kolom ini adalah:
    - NIP (Nomor Induk Pegawai)
    - NDep (Nama Depan)
    - NBlk (Nama Belakang)
    - JK (Jenis Kelamin)
    - Alamat (Alamat)
    - Telp (Telepon)
    - Jabatan (Jabatan)
    - Gaji (Gaji)
    - NoCab (Nomor Cabang)
- *VALUES*:
  - Menunjukkan data yang akan dimasukkan ke dalam tabel. Data untuk setiap baris harus sesuai dengan urutan kolom yang disebutkan sebelumnya.
- *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: 5250000

    - NoCab: 'C101'
- Baris kedua dan seterusnya mengikuti pola yang sama, dengan data yang berbeda.

Hasil :

“assets/Screenshot (57).png” could not be found.

Latihan-1

**GAMBAR :

“assets/Screenshot (34).png” could not be found.

STRIKTUR :

```
SELECT COUNT(NIP) AS JumlahPegawai, COUNT(Jabatan) AS JumlahJabatan FROM pegawai;
```

PENJELASAN :

1. *SELECT COUNT(NIP) AS JumlahPegawai*:
    - *COUNT(NIP)*: Menghitung jumlah baris atau records dalam tabel berdasarkan kolom NIP. Kolom NIP biasanya merupakan nomor identifikasi pegawai.
    - *AS JumlahPegawai*: Memberikan alias JumlahPegawai pada hasil hitungan ini. Alias ini digunakan untuk memberikan nama yang lebih deskriptif pada kolom hasil query.
2. *COUNT(Jabatan) AS JumlahJabatan*:
    - *COUNT(Jabatan)*: Menghitung jumlah baris atau records dalam tabel berdasarkan kolom Jabatan. Kolom Jabatan menunjukkan jenis jabatan pegawai.
    - *AS JumlahJabatan*: Memberikan alias JumlahJabatan pada hasil hitungan ini. Alias ini digunakan untuk memberikan nama yang lebih deskriptif pada kolom hasil query.
3. *FROM pegawai*:
    - Menunjukkan bahwa data yang akan diambil berasal dari tabel pegawai.

Query ini akan menghasilkan dua kolom dalam hasilnya:

- *JumlahPegawai*: Jumlah total pegawai yang terdaftar dalam tabel pegawai.
- *JumlahJabatan*: Jumlah total entri yang ada dalam kolom Jabatan, yang umumnya sama dengan jumlah pegawai jika setiap pegawai memiliki satu jabatan.

Latihan-2

**GAMBAR :

“assets/Screenshot (35).png” could not be found.

STRIKTUR :

```
select count(NIP) AS JumlahPegawai
    -> FROM pegawai
    -> WHERE NoCab = 'C102';
```

PENJELASAN :

- SELECT COUNT(NIP) AS JumlahPegawai: Bagian ini menghitung jumlah baris atau records dalam tabel pegawai berdasarkan kolom NIP. Alias JumlahPegawai digunakan untuk memberikan nama pada hasil hitungan ini, yang akan muncul sebagai kolom dalam hasil yang dikembalikan.
- FROM pegawai: Bagian ini menentukan tabel pegawai sebagai sumber data yang akan di-query. Ini berarti data yang digunakan dalam query berasal dari tabel pegawai.
- WHERE NoCab = 'C102': Bagian ini menetapkan kondisi filter, yaitu hanya menghitung pegawai yang memiliki nilai NoCab sama dengan 'C102'. NoCab mungkin adalah kolom yang menunjukkan kode cabang tempat pegawai bekerja.

Latihan-3

**GAMBAR :

“assets/Screenshot (39).png” could not be found.

STRIKTUR :

```
SELECT SUM(Gaji) AS Gaji_Manager
    -> FROM pegawai
    -> WHERE Jabatan = 'Manager';
```

PENJELASAN :

- SELECT SUM(Gaji) AS Gaji_Manager: Bagian ini memulai query dengan menentukan bahwa kita ingin menghitung jumlah total gaji (SUM(Gaji)) dari semua baris yang memenuhi kondisi tertentu. AS Gaji_Manager memberikan alias Gaji_Manager untuk hasil hitungan ini, yang akan digunakan sebagai nama kolom dalam hasil yang dikembalikan.
- FROM pegawai: Bagian ini menentukan tabel pegawai sebagai sumber data yang akan di-query. Dengan demikian, query akan mencari data gaji dari tabel ini.

- WHERE Jabatan = 'Manager': Bagian ini menetapkan kondisi filter di mana hanya baris yang memiliki nilai Jabatan sama dengan 'Manager' yang akan dimasukkan dalam perhitungan. Ini berarti hanya pegawai yang memiliki jabatan 'Manager' yang akan dihitung total gajinya.

Latihan-4

**GAMBAR :

“assets/Screenshot (40).png” could not be found.

STRIKTUR :

```
SELECT NoCab, COUNT(NIP) AS Jumlah_pegawai
    -> FROM pegawai
    -> GROUP BY NoCab;
```

PENJELASAN :

- *SELECT NoCab, COUNT(NIP) AS Jumlah_pegawai*:
  - *NoCab*: Memilih kolom NoCab yang biasanya menunjukkan kode cabang tempat pegawai bekerja.
  - *COUNT(NIP)*: Menghitung jumlah baris atau records yang memiliki nilai pada kolom NIP. Karena NIP merupakan kolom yang wajib ada untuk setiap pegawai, ini menghitung jumlah pegawai.
  - *AS Jumlah_pegawai*: Memberikan alias Jumlah_pegawai pada hasil hitungan ini. Alias ini akan digunakan sebagai nama kolom dalam hasil query.
- *FROM pegawai*:
  - Menunjukkan bahwa data yang akan diambil berasal dari tabel pegawai.
- *GROUP BY NoCab*:
  - Mengelompokkan hasil query berdasarkan nilai dalam kolom NoCab. Ini memastikan bahwa penghitungan COUNT(NIP) dilakukan secara terpisah untuk setiap nilai unik di kolom NoCab.

Latihan-5

**GAMBAR :

“assets/Screenshot (42).png” could not be found.

STRIKTUR :

```
SELECT NoCab, COUNT(NIP) AS Jumlah_pegawai
    -> FROM pegawai
    -> GROUP BY NoCab HAVING COUNT(NIP) >= 3;
```

PENJELASAN :

- *SELECT NoCab, COUNT(NIP) AS Jumlah_pegawai*:
  - *NoCab*: Memilih kolom NoCab yang biasanya menunjukkan kode cabang tempat pegawai bekerja.
  - *COUNT(NIP)*: Menghitung jumlah baris atau records berdasarkan kolom NIP. Karena NIP biasanya merupakan identifikasi unik untuk setiap pegawai, ini menghitung jumlah pegawai.
  - *AS Jumlah_pegawai*: Memberikan alias Jumlah_pegawai pada hasil hitungan ini, untuk memberi nama yang lebih deskriptif pada kolom hasil.
- *FROM pegawai*:
  - Menunjukkan bahwa data yang akan diambil berasal dari tabel pegawai.
- *GROUP BY NoCab*:
  - Mengelompokkan hasil berdasarkan nilai dalam kolom NoCab. Ini memastikan bahwa penghitungan COUNT(NIP) dilakukan secara terpisah untuk setiap cabang.
- *HAVING COUNT(NIP) >= 3*:
  - *HAVING*: Digunakan untuk menetapkan kondisi pada kelompok data setelah pengelompokan dilakukan. Berbeda dengan WHERE yang digunakan sebelum pengelompokan, HAVING berlaku setelah GROUP BY.
  - *COUNT(NIP) >= 3*: Menyaring kelompok cabang yang memiliki jumlah pegawai lebih dari atau sama dengan tiga. Hanya cabang-cabang yang memenuhi kondisi ini yang akan ditampilkan dalam hasil query.

Latihan-6

**GAMBAR :*

“assets/Screenshot (43).png” could not be found.

STRIKTUR :*

```
SELECT SUM(Gaji) AS Total_Gaji
    -> FROM pegawai;
```

PENJELASAN :

- *SELECT SUM(Gaji) AS Total_Gaji*:
  - *SUM(Gaji)*: Menghitung jumlah total dari semua nilai dalam kolom Gaji. Fungsi agregat SUM() menjumlahkan nilai-nilai dalam kolom tersebut.
  - *AS Total_Gaji*: Memberikan alias Total_Gaji pada hasil hitungan ini. Alias ini akan digunakan sebagai nama kolom dalam hasil query.
- *FROM pegawai*:
  - Menunjukkan bahwa data yang akan diambil berasal dari tabel pegawai.

Latihan-7

**GAMBAR :

“assets/Screenshot (44).png” could not be found.

STRIKTUR :

```
SELECT SUM(Gaji) AS Gaji_Manager
    -> FROM pegawai
    -> WHERE Jabatan = 'Manager';
```

PENJELASAN :

- *SELECT SUM(Gaji) AS Gaji_Manager*:
  - *SUM(Gaji)*: Menghitung jumlah total dari nilai-nilai dalam kolom Gaji. Fungsi agregat SUM() digunakan untuk menjumlahkan semua nilai gaji yang memenuhi kondisi tertentu.
  - *AS Gaji_Manager*: Memberikan alias Gaji_Manager pada hasil hitungan ini. Alias ini akan digunakan sebagai nama kolom dalam hasil query, memberikan deskripsi yang jelas bahwa nilai tersebut adalah total gaji untuk jabatan 'Manager'.
- *FROM pegawai*:
  - Menunjukkan bahwa data yang akan diambil berasal dari tabel pegawai.
- *WHERE Jabatan = 'Manager'*:
  - Menetapkan kondisi filter untuk query. Hanya baris yang memiliki nilai Jabatan sama dengan 'Manager' yang akan dihitung. Dengan kata lain, hanya pegawai dengan jabatan 'Manager' yang akan dihitung total gajinya.

Latihan-8

**GAMBAR :

“assets/Screenshot (45).png” could not be found.

STRIKTUR :

```
SELECT NoCab, SUM(Gaji) AS TotalGaji
    -> FROM pegawai
    -> GROUP BY NoCab;
```

PENJELASAN :

- *SELECT NoCab, SUM(Gaji) AS TotalGaji*:
  - *NoCab*: Memilih kolom NoCab, yang biasanya menunjukkan kode cabang tempat pegawai bekerja.
  - *SUM(Gaji)*: Menghitung jumlah total dari nilai-nilai dalam kolom Gaji. Fungsi agregat SUM() digunakan untuk menjumlahkan gaji dari semua pegawai dalam setiap cabang.
  - *AS TotalGaji*: Memberikan alias TotalGaji pada hasil hitungan ini, yang akan digunakan sebagai nama kolom dalam hasil query.
- *FROM pegawai*:
  - Menunjukkan bahwa data yang akan diambil berasal dari tabel pegawai.
- *GROUP BY NoCab*:
  - Mengelompokkan hasil berdasarkan nilai dalam kolom NoCab. Ini memastikan bahwa perhitungan SUM(Gaji) dilakukan secara terpisah untuk setiap nilai unik di kolom NoCab, yaitu setiap cabang.

Latihan-9

**GAMBAR :

“assets/Screenshot (46).png” could not be found.

STRIKTUR :
sql
SELECT NoCab, SUM(Gaji) AS Total_Gaji
-> FROM pegawai
-> GROUP BY NoCab HAVING SUM(Gaji) >= 8000000;

PENJELASAN :

- *SELECT NoCab, SUM(Gaji) AS Total_Gaji*:

    - *NoCab*: Memilih kolom NoCab, yang biasanya menunjukkan kode cabang tempat pegawai bekerja.
    - *SUM(Gaji)*: Menghitung jumlah total dari nilai-nilai dalam kolom Gaji. Fungsi agregat SUM() digunakan untuk menjumlahkan gaji dari semua pegawai dalam setiap cabang.
    - *AS Total_Gaji*: Memberikan alias Total_Gaji pada hasil hitungan ini, yang akan digunakan sebagai nama kolom dalam hasil query.
- *FROM pegawai*:
    - Menunjukkan bahwa data yang akan diambil berasal dari tabel pegawai.
- *GROUP BY NoCab*:
    - Mengelompokkan hasil berdasarkan nilai dalam kolom NoCab. Ini memastikan bahwa perhitungan SUM(Gaji) dilakukan secara terpisah untuk setiap cabang.
- *HAVING SUM(Gaji) >= 8000000*:
    - *HAVING*: Digunakan untuk menetapkan kondisi pada kelompok data setelah pengelompokan dilakukan. Ini berbeda dari WHERE yang berlaku sebelum pengelompokan.
    - *SUM(Gaji) >= 8000000*: Menyaring hasil agar hanya cabang-cabang yang memiliki total gaji 8.000.000 atau lebih yang ditampilkan.

Latihan-10

**GAMBAR :

“assets/Screenshot (47).png” could not be found.

STRIKTUR :

```
SELECT AVG(Gaji) AS Rata_rata
    -> FROM pegawai;
```

PENJELASAN :

- *SELECT AVG(Gaji) AS Rata_rata*:
    - *AVG(Gaji)*: Menghitung rata-rata dari nilai-nilai dalam kolom Gaji. Fungsi agregat AVG() digunakan untuk menghitung nilai rata-rata dari kolom tersebut.
    - *AS Rata_rata*: Memberikan alias Rata_rata pada hasil perhitungan ini. Alias ini akan digunakan sebagai nama kolom dalam hasil query, memberikan deskripsi yang jelas bahwa nilai tersebut adalah rata-rata gaji.
- *FROM pegawai*:

- Menunjukkan bahwa data yang akan diambil berasal dari tabel pegawai.

Latihan-11

**GAMBAR :

“assets/Screenshot (48).png” could not be found.

STRIKTUR :

```
SELECT AVG(Gaji) AS GajiRataMgr
    -> FROM pegawai
    -> WHERE Jabatan = 'Manager';
```

PENJELASAN :

- *SELECT AVG(Gaji) AS GajiRataMgr*:
  - *AVG(Gaji)*: Menghitung rata-rata nilai dari kolom Gaji. Fungsi agregat AVG() digunakan untuk menghitung nilai rata-rata gaji.
  - *AS GajiRataMgr*: Memberikan alias GajiRataMgr pada hasil perhitungan ini. Alias ini akan digunakan sebagai nama kolom dalam hasil query, memberikan deskripsi yang jelas bahwa nilai tersebut adalah rata-rata gaji untuk jabatan 'Manager'.
- *FROM pegawai*:
  - Menunjukkan bahwa data yang akan diambil berasal dari tabel pegawai.
- *WHERE Jabatan = 'Manager'*:
  - Menetapkan kondisi filter sehingga hanya pegawai yang memiliki nilai Jabatan sama dengan 'Manager' yang dihitung. Ini berarti hanya gaji dari pegawai dengan jabatan 'Manager' yang akan dimasukkan dalam perhitungan rata-rata.

Latiahan-12

**GAMBAR :

“assets/Screenshot (49).png” could not be found.

STRIKTUR :

```
SELECT NoCab, AVG(Gaji) AS RataGaji
    -> FROM pegawai
    -> GROUP BY NoCab;
```

PENJELASAN :

- *SELECT NoCab, AVG(Gaji) AS RataGaji*:
  - *SELECT*: Digunakan untuk menentukan kolom mana yang ingin ditampilkan dalam hasil query.
  - *NoCab*: Kolom pertama yang akan ditampilkan dalam hasil query. Ini adalah kolom yang menunjukkan kode cabang (misalnya, NoCab bisa merujuk pada nomor cabang dari tabel pegawai).
  - *AVG(Gaji) AS RataGaji: Menghitung rata-rata nilai kolom Gaji untuk setiap grup yang dikelompokkan (dalam hal ini, berdasarkan NoCab). Fungsi* AVG *adalah fungsi agregat yang menghitung nilai rata-rata dari kolom yang ditentukan.* AS RataGaji** memberikan alias atau nama lain pada kolom hasil perhitungan rata-rata tersebut, sehingga hasilnya akan ditampilkan dengan nama RataGaji.
- *FROM pegawai*:
  - Menunjukkan tabel dari mana data diambil, dalam hal ini tabel yang bernama pegawai.
- *GROUP BY NoCab*:
  - *GROUP BY*: Digunakan untuk mengelompokkan baris-baris yang memiliki nilai kolom yang sama. Dalam hal ini, data akan dikelompokkan berdasarkan kolom NoCab. Setiap grup berisi baris-baris dengan nilai NoCab yang sama.
  - Setelah pengelompokan, fungsi agregat (seperti AVG) diterapkan pada setiap grup.

Latiahan-13

**GAMBAR :

“assets/Screenshot (51).png” could not be found.

STRIKTUR :

```
SELECT NoCab, AVG(Gaji) AS RataGaji
    -> FROM pegawai
    -> GROUP BY NoCab HAVING NoCab = 'C101' OR NoCab = 'C102';
```

PENJELASAN :

- *SELECT NoCab, AVG(Gaji) AS RataGaji*:
  - *SELECT*: Digunakan untuk menentukan kolom yang ingin ditampilkan dalam hasil query.

    - *NoCab*: Kolom pertama yang ditampilkan dalam hasil query, yang mewakili kode cabang.
    - *AVG(Gaji) AS RataGaji: Fungsi agregat* AVG** digunakan untuk menghitung rata-rata nilai dari kolom Gaji untuk setiap grup yang dibentuk berdasarkan NoCab. Hasilnya diberi nama alias RataGaji.
- *FROM pegawai*:
    - Menunjukkan bahwa data diambil dari tabel pegawai.
- *GROUP BY NoCab*:
    - *GROUP BY* digunakan untuk mengelompokkan data berdasarkan nilai kolom NoCab. Setiap nilai unik dari NoCab akan menjadi satu grup, dan rata-rata gaji (AVG(Gaji)) dihitung untuk masing-masing grup.
- *HAVING NoCab = 'C101' OR NoCab = 'C102'*:
    - *HAVING* adalah klausul yang digunakan untuk memfilter hasil setelah pengelompokan data dengan GROUP BY.
    - Dalam konteks ini, HAVING membatasi hasil query hanya pada grup-grup di mana NoCab adalah 'C101' atau 'C102'.
    - Ini berbeda dari WHERE karena WHERE digunakan sebelum pengelompokan, sedangkan HAVING digunakan setelah pengelompokan.

Latiahan-14

**GAMBAR :

“assets/Screenshot (52).png” could not be found.

STRIKTUR :

```
SELECT MAX(Gaji) AS GajiTerbesar, MIN(Gaji) AS GajiTerkecil
    -> FROM pegawai;
```

PENJELASAN :

- *SELECT MAX(Gaji) AS GajiTerbesar, MIN(Gaji) AS GajiTerkecil*:
    - *MAX(Gaji)*: Fungsi MAX digunakan untuk mencari nilai maksimum dari kolom Gaji. Fungsi ini akan menelusuri semua nilai dalam kolom Gaji dan mengembalikan nilai yang paling besar. Hasil dari fungsi ini akan diberi alias (nama lain) GajiTerbesar.
    - *MIN(Gaji)*: Fungsi MIN digunakan untuk mencari nilai minimum dari kolom Gaji. Fungsi ini akan menelusuri semua nilai dalam kolom Gaji dan mengembalikan nilai yang paling kecil. Hasil dari fungsi ini akan diberi alias (nama lain) GajiTerkecil.

- *FROM pegawai*:
    - *FROM pegawai*: Bagian ini menentukan dari tabel mana data akan diambil. Dalam hal ini, data diambil dari tabel pegawai.

Latiahan-15

**GAMBAR :

“assets/Screenshot (53).png” could not be found.

STRIKTUR :

```
SELECT NoCab, Max(Gaji) AS GajiTerbesar, MIN(Gaji) AS GajiTerkecil
    -> FROM pegawai
    -> GROUP BY NoCab;
```

PENJELASAN :

- *SELECT NoCab, Max(Gaji) AS GajiTerbesar, MIN(Gaji) AS GajiTerkecil*:
    - *NoCab*: Kolom ini akan ditampilkan dalam hasil query untuk menunjukkan nomor cabang.
    - *MAX(Gaji) AS GajiTerbesar*: Fungsi MAX digunakan untuk mencari nilai maksimum dari kolom Gaji dalam setiap grup NoCab. Hasilnya diberi alias GajiTerbesar.
    - *MIN(Gaji) AS GajiTerkecil*: Fungsi MIN digunakan untuk mencari nilai minimum dari kolom Gaji dalam setiap grup NoCab. Hasilnya diberi alias GajiTerkecil.
- *FROM pegawai*:
    - Bagian ini menentukan dari tabel mana data akan diambil. Dalam hal ini, data diambil dari tabel pegawai.
- *GROUP BY NoCab*:
    - *GROUP BY NoCab*: Bagian ini mengelompokkan hasil berdasarkan kolom NoCab. Artinya, data akan dikelompokkan berdasarkan nomor cabang, dan untuk setiap cabang, fungsi MAX dan MIN akan diterapkan pada kolom Gaji.

Latiahan-16

**GAMBAR :

“assets/Screenshot (54).png” could not be found.

STRIKTUR :

```
SELECT NoCab, Max(Gaji) AS GajiTerbesar, MIN(Gaji) AS GajiTerkecil
    -> FROM pegawai
    -> GROUP BY NoCab HAVING COUNT(NIP) >= 3;
```

PENJELASAN :

- *SELECT NoCab, Max(Gaji) AS GajiTerbesar, MIN(Gaji) AS GajiTerkecil*:
  - *NoCab*: Kolom ini menampilkan nomor cabang (NoCab) dalam hasil query.
  - *MAX(Gaji) AS GajiTerbesar*: Fungsi MAX digunakan untuk mencari nilai maksimum dari kolom Gaji dalam setiap grup NoCab. Hasilnya diberi alias GajiTerbesar.
  - *MIN(Gaji) AS GajiTerkecil*: Fungsi MIN digunakan untuk mencari nilai minimum dari kolom Gaji dalam setiap grup NoCab. Hasilnya diberi alias GajiTerkecil.
- *FROM pegawai*:
  - Bagian ini menentukan dari tabel mana data akan diambil. Dalam hal ini, data diambil dari tabel pegawai.
- *GROUP BY NoCab*:
  - *GROUP BY NoCab*: Bagian ini mengelompokkan hasil berdasarkan kolom NoCab. Artinya, data akan dikelompokkan berdasarkan nomor cabang, dan untuk setiap cabang, fungsi MAX dan MIN akan diterapkan pada kolom Gaji.
- *HAVING COUNT(NIP) >= 3*:
  - *HAVING COUNT(NIP) >= 3*: Kondisi ini menyaring hasil grup yang memenuhi kriteria tertentu. Dalam hal ini, hanya cabang yang memiliki jumlah pegawai (dihitung berdasarkan NIP, yang merupakan Nomor Induk Pegawai) sebanyak tiga atau lebih yang akan dimasukkan dalam hasil.

Latiahan-17

**GAMBAR :

“assets/Screenshot (55).png” could not be found.

STRIKTUR :

```
SELECT COUNT(NIP) AS JumlahPegawai, SUM(Gaji) AS TotalGaji,
    -> AVG(Gaji) AS RataGaji, MAX(Gaji) AS GajiMaks, MIN(Gaji) AS GajiMin
    -> FROM pegawai;
```

PENJELASAN :

- *SELECT COUNT(NIP) AS JumlahPegawai*:

    - Bagian ini menghitung jumlah pegawai dalam tabel pegawai dengan cara menghitung berapa banyak data NIP (Nomor Induk Pegawai) yang ada. Hasilnya akan ditampilkan dengan nama alias JumlahPegawai.
- *SUM(Gaji) AS TotalGaji*:
    - Bagian ini menjumlahkan seluruh nilai gaji dari semua pegawai yang ada di tabel pegawai. Hasil penjumlahan ini akan ditampilkan dengan nama alias TotalGaji.
- *AVG(Gaji) AS RataGaji*:
    - Bagian ini menghitung rata-rata gaji dari semua pegawai yang ada di tabel pegawai. Hasil perhitungan rata-rata gaji ini akan ditampilkan dengan nama alias RataGaji.
- *MAX(Gaji) AS GajiMaks*:
    - Bagian ini mencari dan menampilkan gaji tertinggi (maksimum) di antara semua pegawai yang ada di tabel pegawai. Hasilnya akan ditampilkan dengan nama alias GajiMaks.
- *MIN(Gaji) AS GajiMin*:
    - Bagian ini mencari dan menampilkan gaji terendah (minimum) di antara semua pegawai yang ada di tabel pegawai. Hasilnya akan ditampilkan dengan nama alias GajiMin.
- *FROM pegawai*:
    - Bagian ini menunjukkan tabel sumber data yang digunakan, yaitu tabel pegawai. Semua operasi perhitungan (COUNT, SUM, AVG, MAX, MIN) akan dilakukan berdasarkan data yang ada di tabel ini.

Latiahan-18

**GAMBAR :

“assets/Screenshot (56).png” could not be found.

STRIKTUR :

```
SELECT COUNT(NIP) AS Jumlahpegawai, SUM(Gaji) AS TotalGaji,
    -> AVG(Gaji) AS RataGaji, Max(Gaji) AS GajiMks, MIN(Gaji) AS GajiMin
    -> FROM pegawai
    -> WHERE Jabatan = 'Staff' OR Jabatan = 'Sales'
    -> GROUP BY NoCab HAVING SUM(Gaji) <= 2600000;
```

PENJELASAN :

- *SELECT COUNT(NIP) AS Jumlahpegawai*:

    - Bagian ini menghitung jumlah pegawai dengan menghitung jumlah NIP (Nomor Induk Pegawai) yang ada. Hasilnya akan ditampilkan dengan nama alias Jumlahpegawai.
- *SUM(Gaji) AS TotalGaji*:
    - Bagian ini menjumlahkan semua nilai gaji dari pegawai yang memenuhi kriteria. Hasil penjumlahan gaji ini akan ditampilkan dengan nama alias TotalGaji.
- *AVG(Gaji) AS RataGaji*:
    - Bagian ini menghitung rata-rata gaji dari pegawai yang memenuhi kriteria. Hasil perhitungan rata-rata ini akan ditampilkan dengan nama alias RataGaji.
- *MAX(Gaji) AS GajiMks*:
    - Bagian ini mencari dan menampilkan gaji tertinggi (maksimum) di antara pegawai yang memenuhi kriteria. Hasilnya akan ditampilkan dengan nama alias GajiMks.
- *MIN(Gaji) AS GajiMin*:
    - Bagian ini mencari dan menampilkan gaji terendah (minimum) di antara pegawai yang memenuhi kriteria. Hasilnya akan ditampilkan dengan nama alias GajiMin.
- *FROM pegawai*:
    - Bagian ini menunjukkan tabel yang menjadi sumber data, yaitu tabel pegawai.
- *WHERE Jabatan = 'Staff' OR Jabatan = 'Sales'*:
    - Bagian ini menetapkan kondisi untuk memilih hanya pegawai dengan Jabatan yang bernilai 'Staff' atau 'Sales'. Artinya, hanya data pegawai dengan jabatan ini yang akan diproses lebih lanjut dalam query ini.
- *GROUP BY NoCab*:
    - Bagian ini mengelompokkan data berdasarkan kolom NoCab (Nomor Cabang). Artinya, data akan dikelompokkan per cabang, dan hasil perhitungan (jumlah pegawai, total gaji, rata-rata gaji, gaji tertinggi, dan gaji terendah) akan ditampilkan untuk setiap cabang.
- *HAVING SUM(Gaji) <= 2600000*:
    - Bagian ini menetapkan kondisi setelah pengelompokan, yaitu hanya cabang-cabang yang memiliki total gaji kurang dari atau sama dengan 2.600.000 yang akan ditampilkan dalam hasil akhir query. Kondisi ini digunakan setelah pengelompokan karena HAVING digunakan untuk menyaring hasil setelah agregasi (seperti SUM).