

Modul Basis Data V.1.0

Buku panduan ini berisi modul pembangunan basis data. Panduan dikhususkan kepada pemula dalam basis data MySQL yang mana telah mengetahui pengoperasian dasar komputer dan telah mengikuti perkuliahan Basis Data.

KATA PENGANTAR

Assalamualaikum Warahmatullahi Wabarakatuh.

Alhamdulillah puji syukur kehadiran Allah SWT karena rahmatNya yang tak terhingga sehingga Panduan Modul Basis Data dapat dibuat. Sholawat dan salam kehadiran junjungan nabi Muhammad SAW atas petunjuknya sehingga jalan terang terlihat untuk dapat menggapai kebahagiaan dunia dan akhirat.

Semoga Modul ini menjadi bantuan bagi pembacanya yang sedang membutuhkan tambahan keahlian dalam **pembuatan Basis Data**. Tutorial ini telah dirancang untuk pengguna komputer yang ingin mempelajari Basis Data dengan kasus **Presensi dan Penggajian**. Modul ini akan memberikan pemahaman yang cukup tentang Basis Data dengan contoh kasus membuat data presensi dan penggajian karyawan yang mana akan menjadikan tingkat kemampuan dan pemahaman terhadap basis data yang lebih tinggi.

Sebelum memulai dengan modul ini, pembaca diharapkan memiliki pemahaman dasar tentang periferal Komputer seperti mouse, keyboard, monitor, layar, dll. dan operasi dasar dari Komputer.

Banjarbaru, 5 Desember 2021

Tim Pengembang

DAFTAR ISI

KATA PENGANTAR	1
DAFTAR ISI	2
DAFTAR GAMBAR.....	5
PERTEMUAN 1 CRUD.....	7
Pengantar Kasus Presensi dan Penggajian	7
Lokasi	9
Pembuatan Tabel Lokasi.....	9
Insert Data Lokasi.....	10
Select Data Lokasi	10
Update Data Lokasi.....	11
Delete Data Lokasi.....	11
Jabatan	12
Pembuatan Tabel Jabatan	12
Insert Data Jabatan.....	12
Select Data Jabatan	13
Update Data Jabatan	13
Delete Data Jabatan.....	14
Pengguna	14
Pembuatan Tabel Pengguna	14
Insert Data Pengguna	15
Karyawan	15
Pembuatan Tabel Karyawan	15
Insert Data Karyawan	16
Bagian.....	17
Pembuatan Tabel Bagian	17



Insert Data Bagian	18
Jabatan Karyawan.....	18
Pembuatan Tabel Jabatan Karyawan	18
Insert Data Jabatan Karyawan	19
Bagian Karyawan	20
Pembuatan Tabel Bagian Karyawan.....	20
Insert Data Bagian Karyawan.....	20
Penggajian	21
Pembuatan Tabel Penggajian	21
Insert Data Penggajian.....	21
Presensi.....	22
Pembuatan Tabel Presensi	22
Insert Data Presensi	23
PERTEMUAN 2 ALTER DAN SELECT	35
Modifikasi Field.....	35
ADD	35
CHANGE	35
DROP	36
Select & Where.....	36
Wildcard & Like	36
Numeric.....	38
Function.....	39
Select & Count.....	41
PERTEMUAN 3 JOIN.....	43
OneTo Many	43
Many To Many.....	45
PERTEMUAN 4 PENGGAJIAN & PRESENSI	52



Penggajian	52
Jumlah Gaji yang Dibayarkan Per Tahun	52
Rincian Jumlah Gaji yang Dibayarkan Per Tahun	53
Rincian Jumlah Gaji yang Dibayarkan Per Bulan	54
Rincian Jumlah Gaji yang Dibayarkan Per Karyawan dalam 1 Tahun	54
Rincian Bulanan Jumlah Gaji yang Dibayarkan 1 Karyawan dalam 1 Tahun	55
Slip Gaji	56
Presensi.....	57
Jumlah Presensi Seluruh Karyawan per Keterangan Setiap Tahun	57
Rincian Bulanan Jumlah Presensi Seluruh Karyawan per Keterangan	59
Rincian Bulanan Jumlah Presensi 1 Karyawan per Keterangan.....	60
Rincian 1 Bulan Jumlah Presensi 1 Karyawan per Keterangan	61



DAFTAR GAMBAR

Gambar 1 Relasi Tabel	8
Gambar 2 Pembuatan Tabel Lokasi.....	10
Gambar 3 Hasil Tampil Lokasi.....	37
Gambar 4 Hasil Tampil Lokasi dengan Filter	37
Gambar 5 Hasil Tampil Jabatan.....	38
Gambar 6 Hasil Tampil Jabatan dengan Filter	38
Gambar 7 Hasil Tampil Pengguna	39
Gambar 8 Hasil Tampil Login	40
Gambar 9 Hasil Tampil Masa Kerja Karyawan.....	40
Gambar 10 Hasil Tampil Kuantitas Lokasi dengan Filter	41
Gambar 11 Contoh One To Many.....	44
Gambar 12 Hasil Tampil Bagian	44
Gambar 13 Hasil Tampil Lokasi Bagian	45
Gambar 14 Hasil Tampil Jabatan Karyawan	46
Gambar 15 Hasil Tampil Jabatan.....	47
Gambar 16 Hasil Tampil Seluruh Jabatan	48
Gambar 17 Hasil Tampil Jumlah Karyawan pada Jabatan	51
Gambar 18 Hasil Tampil Jumlah Bayar Gaji	53
Gambar 19 Hasil Tampil Rincian Jumlah Gaji yang Dibayarkan Per Tahun	53
Gambar 20 Hasil Tampil Rincian Jumlah Gaji yang Dibayarkan Per Bulan	54
Gambar 21 Hasil Tampil Rincian Jumlah Gaji yang Dibayarkan Per Karyawan dalam 1 Tahun	55
Gambar 22 Rincian Bulanan Jumlah Gaji yang Dibayarkan 1 Karyawan dalam 1 Tahun	56
Gambar 23 Hasil Tampil Slip Gaji	56
Gambar 24 Hasil Tampil Presensi	57
Gambar 25 Jumlah Presensi Seluruh Karyawan per Keterangan Setiap Tahun	58
Gambar 26 Jumlah Presensi Seluruh Karyawan per Keterangan Dalam 1 Tahun.....	59
Gambar 27 Rincian Bulanan Jumlah Presensi Seluruh Karyawan per Keterangan.....	60



Gambar 28 Rincian Bulanan Jumlah Presensi 1 Karyawan per Keterangan.....	61
Gambar 29 Rincian 1 Bulan Jumlah Presensi 1 Karyawan per Keterangan.....	62



PERTEMUAN 1 CRUD

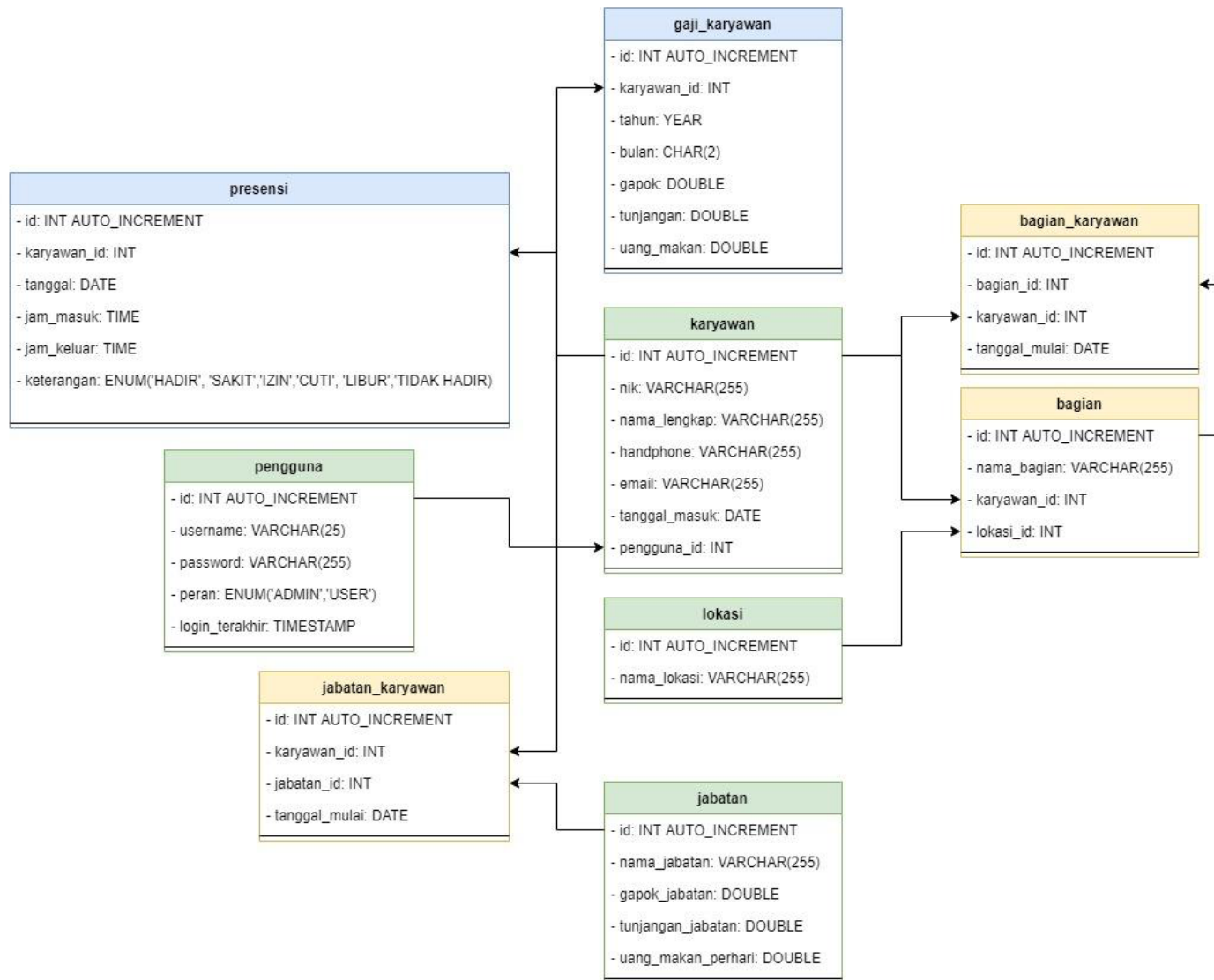
Pengantar Kasus Presensi dan Penggajian

Presensi dan Penggajian merupakan kegiatan yang terdapat di hampir semua instansi. Pada praktek ini disusun sebuah skema dengan data `karyawan` yang sudah diminimalisir, `lokasi` untuk menyimpan data lokasi kantor, `pengguna` yang digunakan untuk menyimpan data username, password, dan peran dari pengguna, `bagian` untuk menyimpan data bagian dan nama Kepala Bagian, `jabatan_karyawan` yang digunakan untuk menyimpan riwayat jabatan karyawan, `bagian_karyawan` digunakan untuk menyimpan data riwayat penempatan karyawan pada suatu bagian, dan tentunya data `presensi` yang menyimpan tanggal presensi, jam masuk, jam keluar, dan keterangan, serta data `gaji_karyawan` yang digunakan untuk menyimpan tahun, bulan, dan jumlah gaji karyawan. Diagram Relasi dapat dilihat pada Gambar 1

Informasi yang bisa dihasilkan dari basis data ini diantaranya:

1. Data seluruh karyawan beserta jabatan terkini, bagian terkini, dan masa kerja
2. Data karyawan beserta riwayat jabatan dan riwayat bagian
3. Menampilkan keterangan presensi semua karyawan setiap tahun, beserta rincian karyawan, bulan, dan tanggalnya
4. Menampilkan jumlah karyawan terlambat setiap tahun, beserta rincian karyawan, bulan, dan tanggalnya
5. Menampilkan slip gaji karyawan
6. Jumlah gaji yang dibayarkan per tahun dan rincian bulannya
7. Jumlah gaji yang diterima per karyawan beserta rinciannya

Basis data ini disusun sedemikian rupa sehingga pembaca dapat mempelajari beragam teknik penyajian data, dan diharapkan dapat membantu memahami konsep dari basis data relasional, sehingga kasus pada dunia nyata tentu saja sedikit banyak berbeda dari kasus yang dipelajari pada modul ini.



Gambar 1 Relasi Tabel



Lokasi

Pembuatan Tabel Lokasi

Tabel `lokasi` digunakan untuk menyimpan data lokasi dan pada modul ini digunakan sebagai pengantar pembuatan tabel sederhana, hanya berisikan 2 (dua) field yaitu `id` dan `nama_lokasi`. Field `id` dibuat sebagai primary key dengan tipe data `INT` dan diberi default value berupa `AUTO_INCREMENT`, sehingga pengembang tidak perlu memikirkan keunikan dari primary key ini, dan akan terus bertambah secara otomatis dikelola penambahannya oleh MySQL. Field `nama_lokasi` menyimpan nama lokasi, dibuat dengan tipe data `varchar` dengan lebar 255 karakter.

Pembuatan tabel lokasi bisa dengan menggunakan perintah SQL berikut:

```
CREATE TABLE lokasi (  
    id INT NOT NULL AUTO_INCREMENT ,  
    nama_lokasi VARCHAR(255) NOT NULL ,  
    PRIMARY KEY (`id`)  
) ENGINE = InnoDB;
```

Terjemahan:

Buat tabel lokasi dengan isi

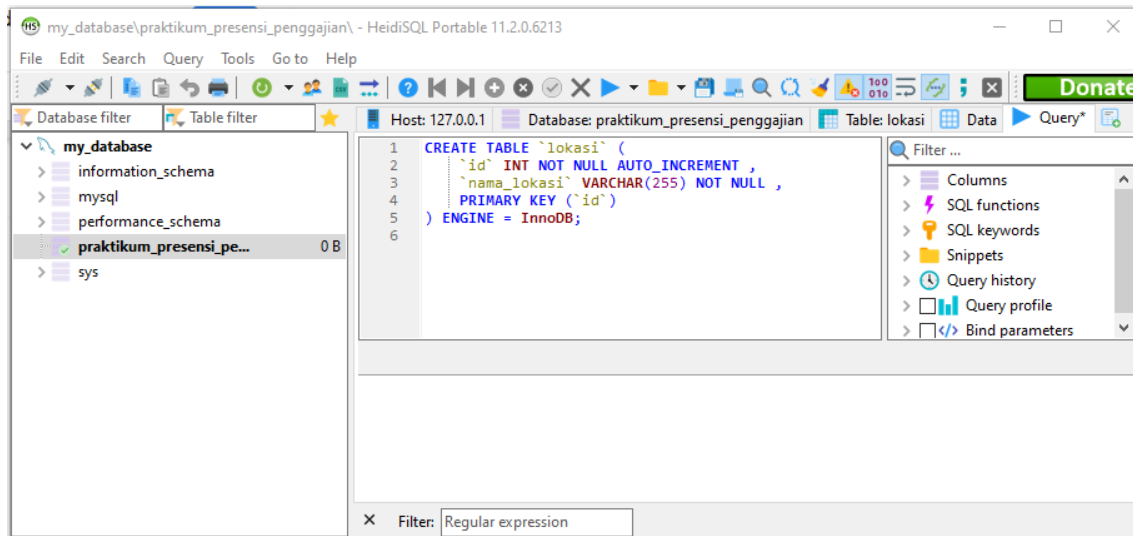
id tipe data int tidak boleh kosong bertambah otomatis,

nama lokasi tipe data varchar lebar 255 tidak boleh kosong,

tetapkan primary key pada field id

gunakan engine InnoDB

Perintah ini bisa dijalankan pada tab `Query` di HeidiSQL. Setelah SQL ditulis bisa dilakukan eksekusi dengan menekan tombol segitiga biru atau bisa disebut dengan Execute SQL (F9), atau klik tombol F9 pada keyboard. Setelah dieksekusi dapat ditampilkan hasil eksekusinya dengan melakukan Refresh (F5).



Gambar 2 Pembuatan Tabel Lokasi

Insert Data Lokasi

Pengisian data lokasi bisa dilakukan dengan menggunakan perintah INSERT pada MySQL. Berikut perintah yang digunakan untuk menambahkan data pada tabel lokasi:

```
INSERT INTO lokasi (id, nama_lokasi) VALUES
(NULL, 'Banjarmasin'),
(NULL, 'Banjarbaru'),
(NULL, 'Gambut');
```

Perintah tersebut bisa dijalankan melalui tab `Query` di HeidiSQL, dan dieksekusi dengan tombol F9. Perintah ini menghasilkan 3 (tiga) data lokasi pada tabel lokasi. Pada field id sengaja diisi dengan NULL karena id bersifat AUTO_INCREMENT dan diatur sendiri oleh MySQL, sedangkan untuk nama_lokasi diisi dengan tipe data string.

Select Data Lokasi

Data lokasi yang sudah diinput bisa ditampilkan dengan menggunakan perintah SELECT pada MySQL. Berikut perintah yang digunakan untuk menambahkan data pada tabel `lokasi`:

```
SELECT * FROM lokasi;
```

Perintah tersebut menampilkan semua field (*) dan semua data dari tabel lokasi.



Update Data Lokasi

Pengubahan data lokasi bisa dilakukan dengan menggunakan perintah `UPDATE` pada MySQL. Berikut perintah yang digunakan untuk mengubah data pada tabel lokasi:

```
UPDATE lokasi SET nama_lokasi = 'Kota Banjarmasin' WHERE
id = 1;
UPDATE lokasi SET nama_lokasi = 'Kota Banjarbaru' WHERE
id = 2;
UPDATE lokasi SET nama_lokasi = 'Kabupaten Banjar' WHERE
id = 3;
```

Perintah tersebut bisa dijalankan melalui tab `Query` di HeidiSQL, dan dieksekusi dengan tombol F9. Perintah ini terdiri dari 3 (tiga) bagian dan bisa dijalankan satu persatu. Perintah pertama adalah mengubah isi field `nama_lokasi` menjadi 'Kota Banjarmasin' dari data pada tabel lokasi yang memiliki field `id` bernilai 1. Perintah kedua dan ketiga melakukan hal yang serupa hanya saja berbeda isi field dan id data yang akan diubah. Perhatikan bahwa kata kunci `WHERE` berperan sangat penting pada perintah ini, jika tidak ada `WHERE` maka semua data pada table akan ikut berubah.

Delete Data Lokasi

Penghapusan data lokasi bisa dilakukan dengan menggunakan perintah `DELETE` pada MySQL. Sebelum menghapus data, bisa dibuat terlebih dahulu data dummy yang akan dihapus, dengan perintah:

```
INSERT INTO lokasi (id, nama_lokasi) VALUES
(NULL, 'Untuk Coba Hapus');
```

Perintah untuk Menghapus data:

```
DELETE FROM lokasi WHERE lokasi.id = 4
```

Perintah tersebut bisa dijalankan melalui tab `Query` di HeidiSQL, dan dieksekusi dengan tombol F9. Perintah ini menghapus data yang memiliki id bernilai 4.

Jabatan

Pembuatan Tabel Jabatan

Tabel `jabatan` digunakan untuk menyimpan data jabatan yang terdapat pada sebuah instansi beserta gaji pokok dan tunjangan yang dibayarkan tiap bulan, serta uang makan perhari yang dibayarkan tiap bulan dan dikalikan dengan jumlah kehadiran karyawan. Field `id` dibuat sebagai primary key dengan tipe data INT dan diberi default value berupa AUTO_INCREMENT, sehingga pengembang tidak perlu memikirkan keunikan dari primary key ini, dan akan terus bertambah secara otomatis dikelola penambahannya oleh MySQL. Tipe data double digunakan untuk field yang berkaitan dengan jumlah uang, untuk memfasilitasi kemungkinan ada operasi pembagian pada field tersebut, seperti pada perhitungan pajak atau lainnya.

Pembuatan tabel `jabatan` bisa dengan menggunakan perintah SQL berikut:

```
CREATE TABLE jabatan (  
    id INT NOT NULL AUTO_INCREMENT ,  
    nama_jabatan VARCHAR(255) NOT NULL ,  
    gapok_jabatan DOUBLE NOT NULL ,  
    tunjangan_jabatan DOUBLE NOT NULL ,  
    uang_makan_perhari DOUBLE NOT NULL ,  
    PRIMARY KEY (id)  
) ENGINE = InnoDB;
```

Perintah ini bisa dijalankan pada tab `Query` di HeidiSQL. Setelah SQL ditulis bisa dilakukan eksekusi dengan menekan tombol segitiga biru atau bisa disebut dengan Execute SQL (F9), atau klik tombol F9 pada keyboard. Setelah dieksekusi dapat ditampilkan hasil eksekusinya dengan melakukan Refresh (F5).

Insert Data Jabatan

Pengisian data jabatan bisa dilakukan dengan menggunakan perintah `INSERT` pada MySQL. Berikut perintah yang digunakan untuk menambahkan data pada tabel `jabatan`:

```
INSERT INTO jabatan (id, nama_jabatan,  
gapok_jabatan, tunjangan_jabatan, uang_makan_perhari)  
VALUES  
(NULL, 'System Analyst', 2400000, 500000, 40000),
```



```
(NULL, 'Project Manager', 2300000, 450000, 38000),  
(NULL, 'Senior Programmer', 2200000, 400000, 36000),  
(NULL, 'Junior Programmer', 2100000, 350000, 34000),  
(NULL, 'Intern', 1000000, 100000, 20000);
```

Perintah tersebut bisa dijalankan melalui tab `Query` di HeidiSQL, dan dieksekusi dengan tombol F9. Perintah ini menghasilkan 5 (lima) data jabatan pada tabel jabatan. Pada field `id` sengaja diisi dengan NULL karena `id` bersifat `AUTO_INCREMENT` dan diatur sendiri oleh MySQL, sedangkan untuk `nama_jabatan` diisi dengan tipe data string menggunakan tanda petik, sedangkan field lainnya yang bertipe data double diisi tanpa menggunakan tanda petik.

Select Data Jabatan

Data jabatan yang sudah diinput bisa ditampilkan dengan menggunakan perintah `SELECT` pada MySQL. Berikut perintah yang digunakan untuk menambahkan data pada tabel jabatan:

```
SELECT * FROM jabatan;
```

Perintah tersebut menampilkan semua field (*) dan semua data dari tabel jabatan.

Update Data Jabatan

Pengubahan data jabatan bisa dilakukan dengan menggunakan perintah `UPDATE` pada MySQL. Berikut perintah yang digunakan untuk mengubah data pada tabel jabatan:

```
UPDATE jabatan SET nama_jabatan = 'Magang' WHERE  
jabatan.id = 5;
```

Perintah tersebut bisa dijalankan melalui tab `Query` di HeidiSQL, dan dieksekusi dengan tombol F9. Perintah ini mengubah isi field `nama_jabatan` menjadi 'Magang' dari data pada tabel jabatan yang memiliki field `id` bernilai 5. Perhatikan bahwa kata kunci `WHERE` berperan sangat penting pada perintah ini, jika tidak ada `WHERE` maka semua data pada table akan ikut berubah.



Delete Data Jabatan

Penghapusan data `jabatan` bisa dilakukan dengan menggunakan perintah `DELETE` pada MySQL. Sebelum menghapus data, bisa dibuat terlebih dahulu data dummy yang akan dihapus, dengan perintah:

```
INSERT INTO jabatan (id, nama_jabatan,  
gapok_jabatan, tunjangan_jabatan, uang_makan_perhari)  
VALUES  
(NULL, 'Untuk Coba Hapus', 999, 999, 999)
```

Perintah untuk Menghapus data:

```
DELETE FROM jabatan WHERE id = 6
```

Perintah tersebut bisa dijalankan melalui tab `Query` di HeidiSQL, dan dieksekusi dengan tombol F9. Perintah ini menghapus data yang memiliki id bernilai 6.

Pengguna

Pembuatan Tabel Pengguna

Tabel `pengguna` digunakan untuk menyimpan data pengguna yang mengakses aplikasi. Tabel `pengguna` memiliki field dengan tipe data `int` untuk `id`, dan `varchar` untuk `username` dan `password`, sedangkan untuk field `peran` menggunakan tipe data `ENUM` dengan isi data yang sudah ditentukan yaitu hanya `ADMIN` dan `USER`. Sedangkan pada field `login_terakhir` menggunakan tipe data `TIMESTAMP` yang pengisian datanya berupa tanggal dan jam yang diisikan langsung oleh MySQL, mirip dengan `INT AUTO INCREMENT` pada `id`.

Pembuatan tabel pengguna bisa dengan menggunakan perintah SQL berikut:

```
CREATE TABLE pengguna (  
  id INT NOT NULL AUTO_INCREMENT,  
  username VARCHAR(255) NOT NULL,  
  password VARCHAR(255) NOT NULL,  
  peran ENUM('ADMIN', 'USER'),  
  login_terakhir TIMESTAMP NULL,  
  PRIMARY KEY (id)  
) ENGINE = InnoDB;
```



Perintah ini bisa dijalankan pada tab `Query` di HeidiSQL. Setelah SQL ditulis bisa dilakukan eksekusi dengan menekan tombol segitiga biru atau bisa disebut dengan Execute SQL (F9), atau klik tombol F9 pada keyboard. Setelah dieksekusi dapat ditampilkan hasil eksekusinya dengan melakukan Refresh (F5).

Insert Data Pengguna

Pengisian data pengguna bisa dilakukan dengan menggunakan perintah `INSERT` pada MySQL. Khusus pada tabel `pengguna` terdapat penambahan fungsi enkripsi MD5 guna melakukan enkripsi pada data `password`, sehingga meskipun data `password` bisa dilihat, namun tidak bisa dengan mudah diketahui `password` sebenarnya yang dituliskan pengguna. Berikut perintah yang digunakan untuk menambahkan data pada tabel `pengguna`:

```
INSERT INTO pengguna (id, username, password,
peran, login_terakhir) VALUES
(NULL, 'admin', MD5('admin'), 'ADMIN', NULL),
(NULL, 'user', MD5('user'), 'USER', NULL),
(NULL, 'johndoe', MD5('johndoe'), 'USER', NULL),
(NULL, 'fulanbinfulan', MD5('fulanbinfulan'), 'USER',
NULL),
(NULL, 'mawar', MD5('mawar'), 'USER', NULL),
(NULL, 'melati', MD5('melati'), 'USER', NULL),
(NULL, 'dahlia', MD5('user'), 'USER', NULL),
(NULL, 'lily', MD5('lily'), 'USER', NULL);
```

Perintah tersebut bisa dijalankan melalui tab Query di HeidiSQL, dan dieksekusi dengan tombol F9. Perintah ini menghasilkan 8 (delapan) data pengguna. Pada field `login_terakhir` sengaja diisi dengan NULL karena memiliki tipe data `TIMESTAMP` yang diatur sendiri oleh MySQL, sedangkan untuk `peran` yang memiliki tipe data `ENUM` diisikan dengan tipe data string menggunakan tanda petik.

Karyawan

Pembuatan Tabel Karyawan

Tabel `karyawan` digunakan untuk menyimpan data karyawan yang mana untuk kasus ini disederhanakan. Tabel `karyawan` akan memiliki relasi ONE TO ONE dengan tabel `pengguna` sehingga terdapat sebuah FOREIGN KEY yaitu field `pengguna_id`.



Hanya ada satu tipe data yang belum ada pada tabel sebelumnya yaitu DATE yang akan digunakan untuk menyimpan data tanggal_masuk karyawan pertama kali bekerja. Sisanya tipe data yang sudah dijelaskan pada penjelasan tabel sebelumnya.

Pembuatan tabel karyawan bisa dengan menggunakan perintah SQL berikut:

```
CREATE TABLE karyawan (  
    id INT NOT NULL AUTO_INCREMENT,  
    nik VARCHAR(255) NOT NULL,  
    nama_lengkap VARCHAR(255) NOT NULL,  
    handphone VARCHAR(255) NOT NULL,  
    email VARCHAR(255) NOT NULL,  
    tanggal_masuk DATE NOT NULL,  
    pengguna_id INT,  
    PRIMARY KEY (id)  
) ENGINE = InnoDB;
```

Perintah ini bisa dijalankan pada tab `Query` di HeidiSQL. Setelah SQL ditulis bisa dilakukan eksekusi dengan menekan tombol segitiga biru atau bisa disebut dengan Execute SQL (F9), atau klik tombol F9 pada keyboard. Setelah dieksekusi dapat ditampilkan hasil eksekusinya dengan melakukan Refresh (F5).

Insert Data Karyawan

Pengisian data karyawan bisa dilakukan dengan menggunakan perintah INSERT pada MySQL. Berikut perintah yang digunakan untuk menambahkan data pada tabel karyawan :

```
INSERT INTO karyawan (id, nik, nama_lengkap,  
handphone, email, tanggal_masuk, pengguna_id  
) VALUES  
(NULL, '001', 'Admin', '08112011',  
'admin@mail.com', '2011-01-01', 1),  
(NULL, '002', 'Tes User', '08112012',  
'user@mail.com', '2012-02-02', 2),  
(NULL, '003', 'John Doe', '08112013',  
'john@mail.com', '2013-03-03', 3),  
(NULL, '004', 'Fulan Bin Fulan', '08112014',  
'fulan@mail.com', '2014-04-04', 4),  
(NULL, '005', 'Mawar Kurniani', '08112015',  
'mawar@mail.com', '2015-05-05', 5),  
(NULL, '006', 'Melati Rahmawati', '08112016',
```



```
'melati@mail.com', '2016-06-06', 6),
(NULL, '007', 'Dahlia Setiani', '08112017',
'dahlia@mail.com', '2017-07-07', 7),
(NULL, '008', 'Lily Handayani', '08112018',
'lily@mail.com', '2018-08-08', 8);
```

Perintah tersebut bisa dijalankan melalui tab Query di HeidiSQL, dan dieksekusi dengan tombol F9. Perintah ini menghasilkan 8 (delapan) data karyawan. Pada field `tanggal_masuk` dengan tipe data DATE di input sebagaimana varchar yaitu menggunakan tanda petik.

Bagian

Pembuatan Tabel Bagian

Tabel `bagian` digunakan untuk menyimpan data bagian instansi yang berisikan `nama_bagian`, FOREIGN KEY dari tabel karyawan yaitu `karyawan_id` dan dari tabel lokasi `lokasi_id`. Pencantuman `karyawan_id` disini dimaksudkan untuk menyimpan karyawan sebagai Kepala Bagian pada bagian tersebut, sedangkan `lokasi_id` merujuk kepada lokasi kantor dari bagian ini.

Pembuatan tabel `bagian` bisa dengan menggunakan perintah SQL berikut:

```
CREATE TABLE bagian (
  id INT NOT NULL AUTO_INCREMENT ,
  nama_bagian VARCHAR(255) NOT NULL ,
  karyawan_id INT NOT NULL ,
  lokasi_id INT NOT NULL ,
  PRIMARY KEY (id)
) ENGINE = InnoDB;
```

Perintah ini bisa dijalankan pada tab `Query` di HeidiSQL. Setelah SQL ditulis bisa dilakukan eksekusi dengan menekan tombol segitiga biru atau bisa disebut dengan Execute SQL (F9), atau klik tombol F9 pada keyboard. Setelah dieksekusi dapat ditampilkan hasil eksekusinya dengan melakukan Refresh (F5).



Insert Data Bagian

Pengisian data bagian bisa dilakukan dengan menggunakan perintah `insert` pada MySQL. Berikut perintah yang digunakan untuk menambahkan data pada tabel bagian:

```
INSERT INTO bagian (id, nama_bagian,  
karyawan_id, lokasi_id) VALUES  
(NULL, 'Autentikasi', 5, 1),  
(NULL, 'Data Science', 3, 1),  
(NULL, 'Backend Developer', 6, 2);
```

Perintah tersebut bisa dijalankan melalui tab Query di HeidiSQL, dan dieksekusi dengan tombol F9. Perintah ini menghasilkan 3 (tiga) data bagian. Sepintas data terlihat seperti tidak bermakna dan susah dipahami karena karyawan dan lokasi hanya berupa id saja. Perintah untuk menampilkan relasi antar tabel yang terlibat akan dijelaskan pada pertemuan berikutnya.

Jabatan Karyawan

Pembuatan Tabel Jabatan Karyawan

Tabel `jabatan_karyawan` digunakan untuk menyimpan data jabatan yang diemban oleh karyawan. Tabel ini merupakan penjabaran dari sifat relasi MANY TO MANY antara tabel karyawan dan jabatan, yang bermakna 1 (satu) karyawan bisa memiliki beberapa jabatan, dalam hal ini digunakan sebagai riwayat jabatan yang pernah diemban, oleh karenanya diperlukan juga field `tanggal_mulai` yang digunakan untuk menyimpan tanggal mulai menjabat, biasanya dalam dokumen resmi disebut Terhitung Mulai Tanggal (TMT). Sedangkan pada sebaliknya 1 (satu) jabatan juga bisa diemban oleh banyak karyawan.

Pembuatan tabel `jabatan_karyawan` bisa dengan menggunakan perintah SQL berikut:

```
CREATE TABLE jabatan_karyawan (  
  id INT NOT NULL AUTO_INCREMENT ,  
  jabatan_id INT NOT NULL ,  
  karyawan_id INT NOT NULL ,  
  tanggal_mulai DATE NOT NULL ,  
  PRIMARY KEY (id)  
) ENGINE = InnoDB;
```



Perintah ini bisa dijalankan pada tab `Query` di HeidiSQL. Setelah SQL ditulis bisa dilakukan eksekusi dengan menekan tombol segitiga biru atau bisa disebut dengan Execute SQL (F9), atau klik tombol F9 pada keyboard. Setelah dieksekusi dapat ditampilkan hasil eksekusinya dengan melakukan Refresh (F5).

Insert Data Jabatan Karyawan

Pengisian data jabatan karyawan bisa dilakukan dengan menggunakan perintah INSERT pada MySQL. Berikut perintah yang digunakan untuk menambahkan data pada tabel jabatan_karyawan:

```
INSERT INTO jabatan_karyawan (id, karyawan_id,
jabatan_id, tanggal_mulai) VALUES
(NULL, 3, 5, '2013-03-03'),
(NULL, 3, 4, '2014-04-01'),
(NULL, 4, 5, '2014-04-04'),
(NULL, 3, 3, '2015-05-04'),
(NULL, 4, 4, '2015-05-05'),
(NULL, 5, 5, '2015-05-05'),
(NULL, 3, 2, '2015-06-01'),
(NULL, 4, 3, '2015-06-02'),
(NULL, 5, 4, '2015-06-03'),
(NULL, 6, 5, '2015-06-06'),
(NULL, 3, 1, '2017-07-01'),
(NULL, 4, 2, '2017-07-02'),
(NULL, 5, 3, '2017-07-02'),
(NULL, 6, 4, '2017-07-02'),
(NULL, 7, 5, '2017-07-07'),
(NULL, 7, 4, '2018-08-02'),
(NULL, 8, 5, '2018-08-08');
```

Perintah tersebut bisa dijalankan melalui tab Query di HeidiSQL, dan dieksekusi dengan tombol F9. Perintah ini menghasilkan 3 (tiga) data bagian. Serupa dengan tabel bagian pada sub bab sebelumnya, pada tabel ini karyawan dan jabatan hanya diwakili oleh id nya saja, dan bisa diamati pada data ada 1 (satu) karyawan yang memiliki beberapa jabatan, begitu juga sebaliknya.

Bagian Karyawan

Pembuatan Tabel Bagian Karyawan

Tabel `bagian_karyawan` digunakan untuk menyimpan data penempatan karyawan pada suatu bagian. Sama seperti `jabatan_karyawan` tabel ini juga memiliki field `tanggal_mulai` sebagai data TMT penempatan karyawan.

Pembuatan tabel `bagian_karyawan` bisa dengan menggunakan perintah SQL berikut:

```
CREATE TABLE bagian_karyawan (  
  id INT NOT NULL AUTO_INCREMENT ,  
  bagian_id INT NOT NULL ,  
  karyawan_id INT NOT NULL ,  
  tanggal_mulai DATE NOT NULL ,  
  PRIMARY KEY (id)  
) ENGINE = InnoDB;
```

Perintah ini bisa dijalankan pada tab `Query` di HeidiSQL. Setelah SQL ditulis bisa dilakukan eksekusi dengan menekan tombol segitiga biru atau bisa disebut dengan Execute SQL (F9), atau klik tombol F9 pada keyboard. Setelah dieksekusi dapat ditampilkan hasil eksekusinya dengan melakukan Refresh (F5).

Insert Data Bagian Karyawan

Pengisian data bagian karyawan bisa dilakukan dengan menggunakan perintah INSERT pada MySQL. Berikut perintah yang digunakan untuk menambahkan data pada tabel `bagian_karyawan`:

```
INSERT INTO `bagian_karyawan` (`id`, `bagian_id`,  
`karyawan_id`, `tanggal_mulai`) VALUES  
(NULL, 1, 5, '2018-03-03'),  
(NULL, 2, 3, '2018-04-01'),  
(NULL, 3, 6, '2018-04-04'),  
(NULL, 1, 4, '2018-03-03'),  
(NULL, 2, 7, '2018-04-01'),  
(NULL, 3, 8, '2018-04-04');
```

Perintah tersebut bisa dijalankan melalui tab Query di HeidiSQL, dan dieksekusi dengan tombol F9.

Penggajian

Pembuatan Tabel Penggajian

Tabel `penggajian` digunakan untuk menyimpan data pembayaran gaji karyawan setiap bulan. Field yang dibuat pada `penggajian` diantaranya `karyawan_id` sebagai FOREIGN KEY dari tabel `karyawan`, tahun dan bulan pembayaran gaji. Field `gapok`, `tunjangan`, dan `uang_makan` disimpan tersendiri meskipun sudah ada pada tabel `jabatan`, hal ini guna mengantisipasi terjadi perubahan gaji pada jabatan, sehingga meskipun ada perubahan besaran nilai gaji pada jabatan, data gaji yang dibayarkan tidak ikut berubah.

Pembuatan tabel `penggajian` bisa dengan menggunakan perintah SQL berikut:

```
CREATE TABLE penggajian (  
  id INT NOT NULL AUTO_INCREMENT ,  
  karyawan_id INT NOT NULL ,  
  tahun YEAR ,  
  bulan CHAR(2) ,  
  gapok DOUBLE ,  
  tunjangan DOUBLE ,  
  uang_makan DOUBLE ,  
  PRIMARY KEY (id)  
) ENGINE = InnoDB;
```

Perintah ini bisa dijalankan pada tab `Query` di HeidiSQL. Setelah SQL ditulis bisa dilakukan eksekusi dengan menekan tombol segitiga biru atau bisa disebut dengan Execute SQL (F9), atau klik tombol F9 pada keyboard. Setelah dieksekusi dapat ditampilkan hasil eksekusinya dengan melakukan Refresh (F5).

Insert Data Penggajian

Pengisian data `penggajian` bisa dilakukan dengan menggunakan perintah `INSERT` pada MySQL. Berikut perintah yang digunakan untuk menambahkan data pada tabel `penggajian`:

```
INSERT INTO penggajian (id, karyawan_id, tahun,  
  bulan, gapok, tunjangan, uang_makan) VALUES
```



```
(NULL, 3, '2020', '11', 2400000, 500000, 800000),
(NULL, 4, '2020', '11', 2300000, 450000, 684000),
(NULL, 5, '2020', '11', 2200000, 400000, 684000),
(NULL, 6, '2020', '11', 2100000, 350000, 640000),
(NULL, 7, '2020', '11', 2100000, 350000, 612000),
(NULL, 8, '2020', '11', 1000000, 100000, 380000),
(NULL, 3, '2020', '12', 2400000, 500000, 800000),
(NULL, 4, '2020', '12', 2300000, 450000, 722000),
(NULL, 5, '2020', '12', 2200000, 400000, 720000),
(NULL, 6, '2020', '12', 2100000, 350000, 680000),
(NULL, 7, '2020', '12', 2100000, 350000, 646000),
(NULL, 8, '2020', '12', 1000000, 100000, 360000),
(NULL, 3, '2021', '01', 2400000, 500000, 800000),
(NULL, 4, '2021', '01', 2300000, 450000, 722000),
(NULL, 5, '2021', '01', 2200000, 400000, 612000),
(NULL, 6, '2021', '01', 2100000, 350000, 680000),
(NULL, 7, '2021', '01', 2100000, 350000, 646000),
(NULL, 8, '2021', '01', 1000000, 100000, 340000);
```

Perintah tersebut bisa dijalankan melalui tab Query di HeidiSQL, dan dieksekusi dengan tombol F9.

Presensi

Pembuatan Tabel Presensi

Tabel `presensi` digunakan untuk menyimpan data kehadiran karyawan setiap hari dengan mencatat ``jam_masuk``, ``jam_keluar`` dan ``keterangan`` kehadiran ataupun ketidakhadiran.

Pembuatan tabel `presensi` bisa dengan menggunakan perintah SQL berikut:

```
CREATE TABLE presensi (
  id INT NOT NULL AUTO_INCREMENT ,
  karyawan_id INT NOT NULL ,
  tanggal DATE NOT NULL ,
  jam_masuk TIME ,
  jam_keluar TIME ,
  keterangan ENUM('HADIR','SAKIT','IZIN','CUTI','AKHIR
PEKAN','LIBUR NASIONAL','TANPA KETERANGAN'),
  PRIMARY KEY (id)
) ENGINE = InnoDB;
```



Perintah ini bisa dijalankan pada tab `Query` di HeidiSQL. Setelah SQL ditulis bisa dilakukan eksekusi dengan menekan tombol segitiga biru atau bisa disebut dengan Execute SQL (F9), atau klik tombol F9 pada keyboard. Setelah dieksekusi dapat ditampilkan hasil eksekusinya dengan melakukan Refresh (F5).

Insert Data Presensi

Pengisian data bagian bisa dilakukan dengan menggunakan perintah INSERT pada MySQL. Data presensi bisa disalin dari [halaman ini \(klik\)](#) Berikut perintah yang digunakan untuk menambahkan data pada tabel presensi :

```
INSERT INTO presensi (id, karyawan_id, tanggal, jam_masuk, jam_keluar, keterangan) VALUES
(NULL, 3, '2020-11-01', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 3, '2020-11-02', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-03', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-04', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-05', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-06', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-07', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 3, '2020-11-08', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 3, '2020-11-09', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-10', NULL, NULL, 'LIBUR NASIONAL'),
(NULL, 3, '2020-11-11', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-12', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-13', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-14', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 3, '2020-11-15', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 3, '2020-11-16', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-17', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-18', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-19', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-20', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-21', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 3, '2020-11-22', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 3, '2020-11-23', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-24', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-25', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-26', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-27', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-11-28', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 3, '2020-11-29', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 3, '2020-11-30', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-12-01', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2020-12-02', '08:00', '16:00', 'HADIR'),
```


(NULL, 3, '2020-12-03', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-04', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-05', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 3, '2020-12-06', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 3, '2020-12-07', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-08', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-09', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 3, '2020-12-10', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-11', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-12', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 3, '2020-12-13', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 3, '2020-12-14', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-15', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-16', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-17', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-18', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-19', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 3, '2020-12-20', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 3, '2020-12-21', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-22', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-23', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-24', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 3, '2020-12-25', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 3, '2020-12-26', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 3, '2020-12-27', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 3, '2020-12-28', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-29', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-30', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2020-12-31', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2021-01-01', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 3, '2021-01-02', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 3, '2021-01-03', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 3, '2021-01-04', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2021-01-05', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2021-01-06', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2021-01-07', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2021-01-08', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2021-01-09', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 3, '2021-01-10', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 3, '2021-01-11', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2021-01-12', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2021-01-13', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2021-01-14', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2021-01-15', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2021-01-16', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 3, '2021-01-17', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 3, '2021-01-18', '08:00', '16:00', 'HADIR'),
 (NULL, 3, '2021-01-19', '08:00', '16:00', 'HADIR'),

```

(NULL, 3, '2021-01-20', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2021-01-21', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2021-01-22', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2021-01-23', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 3, '2021-01-24', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 3, '2021-01-25', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2021-01-26', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2021-01-27', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2021-01-28', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2021-01-29', '08:00', '16:00', 'HADIR'),
(NULL, 3, '2021-01-30', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 3, '2021-01-31', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 4, '2020-11-01', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 4, '2020-11-02', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-11-03', '08:35', '16:00', 'HADIR'),
(NULL, 4, '2020-11-04', NULL, NULL, 'SAKIT'),
(NULL, 4, '2020-11-05', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-11-06', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-11-07', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 4, '2020-11-08', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 4, '2020-11-09', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-11-10', NULL, NULL, 'LIBUR NASIONAL'),
(NULL, 4, '2020-11-11', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-11-12', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-11-13', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-11-14', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 4, '2020-11-15', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 4, '2020-11-16', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-11-17', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-11-18', '08:45', '16:00', 'HADIR'),
(NULL, 4, '2020-11-19', NULL, NULL, 'IZIN'),
(NULL, 4, '2020-11-20', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-11-21', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 4, '2020-11-22', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 4, '2020-11-23', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-11-24', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-11-25', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-11-26', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-11-27', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-11-28', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 4, '2020-11-29', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 4, '2020-11-30', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-12-01', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-12-02', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-12-03', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-12-04', '08:00', '16:00', 'HADIR'),
(NULL, 4, '2020-12-05', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 4, '2020-12-06', NULL, NULL, 'AKHIR PEKAN'),

```

(NULL, 4, '2020-12-07', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2020-12-08', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2020-12-09', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 4, '2020-12-10', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2020-12-11', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2020-12-12', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 4, '2020-12-13', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 4, '2020-12-14', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2020-12-15', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2020-12-16', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2020-12-17', NULL, NULL, 'SAKIT'),
 (NULL, 4, '2020-12-18', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2020-12-19', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 4, '2020-12-20', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 4, '2020-12-21', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2020-12-22', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2020-12-23', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2020-12-24', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 4, '2020-12-25', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 4, '2020-12-26', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 4, '2020-12-27', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 4, '2020-12-28', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2020-12-29', '08:50', '16:00', 'HADIR'),
 (NULL, 4, '2020-12-30', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2020-12-31', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-01', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 4, '2021-01-02', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 4, '2021-01-03', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 4, '2021-01-04', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-05', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-06', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-07', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-08', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-09', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 4, '2021-01-10', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 4, '2021-01-11', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-12', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-13', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-14', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-15', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-16', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 4, '2021-01-17', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 4, '2021-01-18', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-19', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-20', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-21', NULL, NULL, 'IZIN'),
 (NULL, 4, '2021-01-22', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-23', NULL, NULL, 'AKHIR PEKAN'),

(NULL, 4, '2021-01-24', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 4, '2021-01-25', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-26', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-27', '08:40', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-28', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-29', '08:00', '16:00', 'HADIR'),
 (NULL, 4, '2021-01-30', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 4, '2021-01-31', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-11-01', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-11-02', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-03', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-04', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-05', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-06', '08:35', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-07', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-11-08', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-11-09', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-10', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 5, '2020-11-11', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-12', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-13', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-14', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-11-15', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-11-16', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-17', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-18', '08:35', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-19', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-20', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-21', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-11-22', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-11-23', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-24', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-25', NULL, NULL, 'SAKIT'),
 (NULL, 5, '2020-11-26', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-27', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-11-28', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-11-29', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-11-30', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-01', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-02', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-03', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-04', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-05', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-12-06', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-12-07', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-08', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-09', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 5, '2020-12-10', '08:00', '16:00', 'HADIR'),

(NULL, 5, '2020-12-11', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-12', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-12-13', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-12-14', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-15', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-16', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-17', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-18', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-19', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-12-20', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-12-21', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-22', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-23', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-24', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 5, '2020-12-25', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 5, '2020-12-26', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-12-27', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2020-12-28', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-29', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-30', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2020-12-31', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-01', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 5, '2021-01-02', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2021-01-03', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2021-01-04', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-05', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-06', NULL, NULL, 'IZIN'),
 (NULL, 5, '2021-01-07', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-08', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-09', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2021-01-10', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2021-01-11', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-12', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-13', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-14', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-15', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-16', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2021-01-17', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2021-01-18', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-19', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-20', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-21', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-22', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-23', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2021-01-24', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2021-01-25', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-26', NULL, NULL, 'SAKIT'),
 (NULL, 5, '2021-01-27', NULL, NULL, 'SAKIT'),

(NULL, 5, '2021-01-28', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-29', '08:00', '16:00', 'HADIR'),
 (NULL, 5, '2021-01-30', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 5, '2021-01-31', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-11-01', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-11-02', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-03', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-04', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-05', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-06', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-07', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-11-08', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-11-09', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-10', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 6, '2020-11-11', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-12', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-13', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-14', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-11-15', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-11-16', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-17', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-18', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-19', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-20', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-21', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-11-22', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-11-23', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-24', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-25', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-26', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-27', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-11-28', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-11-29', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-11-30', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-01', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-02', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-03', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-04', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-05', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-12-06', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-12-07', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-08', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-09', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 6, '2020-12-10', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-11', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-12', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-12-13', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-12-14', '08:00', '16:00', 'HADIR'),

(NULL, 6, '2020-12-15', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-16', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-17', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-18', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-19', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-12-20', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-12-21', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-22', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-23', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-24', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 6, '2020-12-25', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 6, '2020-12-26', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-12-27', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2020-12-28', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-29', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-30', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2020-12-31', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-01', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 6, '2021-01-02', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2021-01-03', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2021-01-04', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-05', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-06', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-07', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-08', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-09', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2021-01-10', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2021-01-11', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-12', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-13', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-14', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-15', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-16', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2021-01-17', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2021-01-18', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-19', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-20', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-21', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-22', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-23', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2021-01-24', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2021-01-25', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-26', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-27', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-28', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-29', '08:00', '16:00', 'HADIR'),
 (NULL, 6, '2021-01-30', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 6, '2021-01-31', NULL, NULL, 'AKHIR PEKAN'),

(NULL, 7, '2020-11-01', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-11-02', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-03', '08:35', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-04', NULL, NULL, 'SAKIT'),
 (NULL, 7, '2020-11-05', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-06', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-07', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-11-08', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-11-09', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-10', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 7, '2020-11-11', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-12', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-13', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-14', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-11-15', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-11-16', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-17', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-18', '08:45', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-19', NULL, NULL, 'IZIN'),
 (NULL, 7, '2020-11-20', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-21', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-11-22', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-11-23', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-24', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-25', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-26', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-27', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-11-28', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-11-29', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-11-30', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-01', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-02', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-03', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-04', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-05', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-12-06', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-12-07', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-08', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-09', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 7, '2020-12-10', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-11', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-12', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-12-13', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-12-14', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-15', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-16', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-17', NULL, NULL, 'SAKIT'),
 (NULL, 7, '2020-12-18', '08:00', '16:00', 'HADIR'),



(NULL, 7, '2020-12-19', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-12-20', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-12-21', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-22', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-23', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-24', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 7, '2020-12-25', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 7, '2020-12-26', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-12-27', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2020-12-28', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-29', '08:50', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-30', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2020-12-31', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-01', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 7, '2021-01-02', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2021-01-03', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2021-01-04', '08:55', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-05', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-06', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-07', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-08', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-09', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2021-01-10', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2021-01-11', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-12', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-13', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-14', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-15', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-16', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2021-01-17', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2021-01-18', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-19', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-20', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-21', NULL, NULL, 'IZIN'),
 (NULL, 7, '2021-01-22', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-23', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2021-01-24', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2021-01-25', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-26', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-27', '08:40', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-28', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-29', '08:00', '16:00', 'HADIR'),
 (NULL, 7, '2021-01-30', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 7, '2021-01-31', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 8, '2020-11-01', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 8, '2020-11-02', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-03', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-04', '08:00', '16:00', 'HADIR'),

(NULL, 8, '2020-11-05', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-06', '08:35', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-07', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 8, '2020-11-08', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 8, '2020-11-09', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-10', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 8, '2020-11-11', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-12', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-13', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-14', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 8, '2020-11-15', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 8, '2020-11-16', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-17', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-18', '08:35', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-19', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-20', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-21', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 8, '2020-11-22', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 8, '2020-11-23', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-24', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-25', NULL, NULL, 'IZIN'),
 (NULL, 8, '2020-11-26', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-27', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-11-28', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 8, '2020-11-29', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 8, '2020-11-30', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-12-01', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-12-02', NULL, NULL, 'CUTI'),
 (NULL, 8, '2020-12-03', NULL, NULL, 'CUTI'),
 (NULL, 8, '2020-12-04', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-12-05', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 8, '2020-12-06', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 8, '2020-12-07', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-12-08', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-12-09', NULL, NULL, 'LIBUR NASIONAL'),
 (NULL, 8, '2020-12-10', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-12-11', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-12-12', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 8, '2020-12-13', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 8, '2020-12-14', '08:37', '16:00', 'HADIR'),
 (NULL, 8, '2020-12-15', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-12-16', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-12-17', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-12-18', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-12-19', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 8, '2020-12-20', NULL, NULL, 'AKHIR PEKAN'),
 (NULL, 8, '2020-12-21', '08:00', '16:00', 'HADIR'),
 (NULL, 8, '2020-12-22', '08:00', '16:00', 'HADIR'),



```
(NULL, 8, '2020-12-23', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2020-12-24', NULL, NULL, 'LIBUR NASIONAL'),
(NULL, 8, '2020-12-25', NULL, NULL, 'LIBUR NASIONAL'),
(NULL, 8, '2020-12-26', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 8, '2020-12-27', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 8, '2020-12-28', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2020-12-29', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2020-12-30', '08:35', '16:00', 'HADIR'),
(NULL, 8, '2020-12-31', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-01', NULL, NULL, 'LIBUR NASIONAL'),
(NULL, 8, '2021-01-02', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 8, '2021-01-03', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 8, '2021-01-04', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-05', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-06', NULL, NULL, 'IZIN'),
(NULL, 8, '2021-01-07', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-08', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-09', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 8, '2021-01-10', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 8, '2021-01-11', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-12', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-13', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-14', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-15', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-16', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 8, '2021-01-17', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 8, '2021-01-18', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-19', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-20', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-21', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-22', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-23', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 8, '2021-01-24', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 8, '2021-01-25', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-26', NULL, NULL, 'SAKIT'),
(NULL, 8, '2021-01-27', NULL, NULL, 'SAKIT'),
(NULL, 8, '2021-01-28', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-29', '08:00', '16:00', 'HADIR'),
(NULL, 8, '2021-01-30', NULL, NULL, 'AKHIR PEKAN'),
(NULL, 8, '2021-01-31', NULL, NULL, 'AKHIR PEKAN');
```

Perintah tersebut bisa dijalankan melalui tab Query di HeidiSQL, dan dieksekusi dengan tombol F9.



PERTEMUAN 2 ALTER DAN SELECT

Modifikasi Field

ADD

Pada saat mengembangkan sebuah aplikasi, bukan tidak mungkin terjadi berbagai macam perubahan pada struktur basis data. Tidak jarang perubahan ini jauh berbeda dengan perancangannya. Perubahan yang mungkin terjadi pada struktur basis data diantaranya adalah penambahan field, perubahan nama field dan tipe datanya, dan juga penghapusan field. Semua perubahan tersebut bisa dilakukan menggunakan fitur `ALTER TABLE`.

Penambahan field pada sebuah tabel bisa menggunakan perintah `ALTER TABLE ADD`, berikut adalah contoh perubahan pada tabel `lokasi` berupa penambahan field `alamat` dengan tipe data `varchar`, field `alamat` yang akan ditambahkan, diletakkan setelah field `nama_lokasi`.

```
ALTER TABLE lokasi ADD alamat VARCHAR(255) NOT NULL AFTER nama_lokasi;
```

Terjemahan:

Perubahan pada (`ALTER`) tabel (`TABLE lokasi`) berupa penambahan (`ADD`) field (`alamat`) dengan tipe data `varchar` (`VARCHAR(255)`) dan tidak boleh kosong (`NOT NULL`), field `alamat` yang akan ditambahkan, diletakkan setelah (`AFTER`) field (`nama_lokasi`).

CHANGE

Perubahan field bisa dilakukan menggunakan perintah `ALTER TABLE CHANGE`, berikut adalah contoh perubahan pada tabel `lokasi` berupa perubahan nama field `alamat` menjadi `alamat_gedung` dengan tipe data `TEXT` dan dibuat tidak boleh kosong

```
ALTER TABLE lokasi CHANGE alamat alamat_gedung TEXT NOT NULL;
```

Terjemahan:

Perubahan pada (ALTER) tabel lokasi (TABLE lokasi) berupa perubahan (CHANGE) field alamat (alamat) menjadi (alamat_gedung) dengan tipe data text (TEXT) dan tidak boleh kosong (NOT NULL).

DROP

Penghapusan field bisa dilakukan menggunakan perintah ALTER TABLE DROP, berikut adalah contoh penghapusan field alamat_gedung

```
ALTER TABLE lokasi DROP alamat_gedung;
```

Terjemahan:

Perubahan pada (ALTER) tabel lokasi (TABLE lokasi) berupa penghapusan (DROP) field (alamat_gedung).

Select & Where

Select merupakan perintah yang digunakan untuk menampilkan data yang ada pada tabel sebagaimana sudah dicontohkan pada pertemuan pertama. Kata kunci Where bisa ditambahkan pada perintah Select sebagai penyaring (filter) data. Misalkan menampilkan lokasi dengan id=1, atau menampilkan lokasi dengan nama tertentu. Berikut contohnya:

```
SELECT * FROM lokasi WHERE id = 2;
```

Terjemahan:

Pilih (SELECT) semua field (*) dari tabel lokasi (FROM lokasi) dimana id lokasinya adalah 2 (WHERE id = 2)

Wildcard & Like

Perintah SELECT * FROM lokasi Menampilkan seluruh data lokasi seperti yang tampil pada gambar berikut



1	SELECT * FROM lokasi	
---	----------------------	--

lokasi (3r × 2c)	
id	nama_lokasi
1	Kota Banjarmasin
2	Kota Banjarbaru
3	Kabupaten Banjar

Gambar 3 Hasil Tampil Lokasi

Ketika diperlukan data yang nama kotanya memiliki unsur kota maka bisa digunakan perintah berikut

```
SELECT * FROM lokasi WHERE nama_lokasi like '%kota%'
```

Terjemahan:

Pilih (SELECT) semua field (*) dari tabel lokasi (FROM lokasi) dimana id nama lokasinya mirip dengan kota (WHERE nama_lokasi like '%kota%')

1	SELECT * FROM lokasi WHERE nama_lokasi like '%kota%'	
---	--	--

lokasi (2r × 2c)	
id	nama_lokasi
1	Kota Banjarmasin
2	Kota Banjarbaru

Gambar 4 Hasil Tampil Lokasi dengan Filter

Penggunaan simbol *wildcard* (%) jika didahului dengan kata `like` maka akan digunakan sebagai pengganti karakter apapun yang ada pada string tersebut. Misalkan jika nama lokasi `Kota Banjarbaru` maka dianggap bernilai true, contoh lain jika nama lokasi `Terakota` maka dianggap bernilai true juga.

Latihan 2.1

Buat perintah SQL yang menampilkan tabel jabatan dengan hanya menampilkan jabatan yang mengandung kata `programmer`

id	nama_jabatan	gapok_jabatan	tunjangan_jabatan	uang_makan_perhari
3	Senior Programmer	2.200.000	400.000	36.000
4	Junior Programmer	2.100.000	350.000	34.000

Numeric

Kata kunci WHERE bisa diterapkan pada tipe data numerik, baik itu integer maupun double, dan juga bisa menggunakan operator lebih besar dari (>) dan kurang dari (<).

Berikut adalah tampilan data seluruh jabatan

```
1 SELECT * FROM jabatan
```

jabatan (5r x 5c)				
id	nama_jabatan	gapok_jabatan	tunjangan_jabatan	uang_makan_perhari
1	System Analyst	2.400.000	500.000	40.000
2	Project Manager	2.300.000	450.000	38.000
3	Senior Programmer	2.200.000	400.000	36.000
4	Junior Programmer	2.100.000	350.000	34.000
5	Magang	1.000.000	100.000	20.000

Gambar 5 Hasil Tampil Jabatan

Berikut contoh pada data jabatan menampilkan jabatan yang memiliki gaji pokok lebih besar dari 2.200.000

```
1 SELECT * FROM jabatan WHERE gapok_jabatan > 2200000;
```

jabatan (2r x 5c)				
id	nama_jabatan	gapok_jabatan	tunjangan_jabatan	uang_makan_perhari
1	System Analyst	2.400.000	500.000	40.000
2	Project Manager	2.300.000	450.000	38.000

Gambar 6 Hasil Tampil Jabatan dengan Filter



```
SELECT * FROM jabatan WHERE gapok_jabatan > 2200000;
```

Terjemahan:

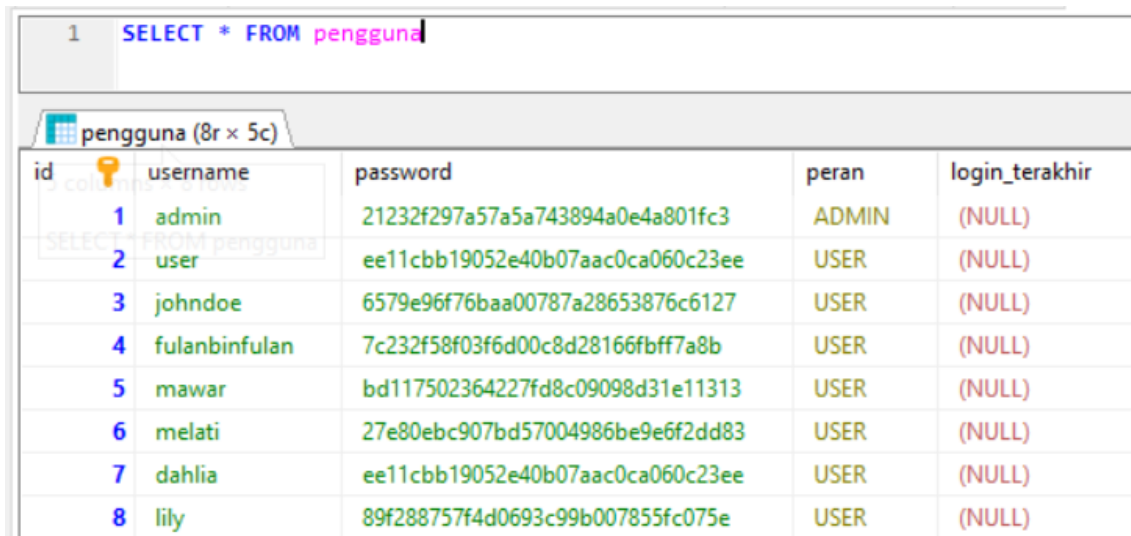
Pilih (SELECT) semua field (*) dari tabel jabatan (FROM jabatan) dimana gaji pokok jabatan lebih besar dari 2.200.000 (Dua juta dua ratus ribu) (WHERE gapok_jabatan > 2200000;)

Latihan 2.2

Buat perintah SQL yang menampilkan tabel jabatan dengan hanya menampilkan jabatan dengan uang makan perhari lebih kecil dari 35.000

Function

Penulisan WHERE bisa juga disertakan dengan fungsi (function) yang disediakan MySQL, seperti fungsi MD5 yang digunakan untuk implementasi login, berikut tampilan seluruh data pengguna



id	username	password	peran	login_terakhir
1	admin	21232f297a57a5a743894a0e4a801fc3	ADMIN	(NULL)
2	user	ee11cbb19052e40b07aac0ca060c23ee	USER	(NULL)
3	johndoe	6579e96f76baa00787a28653876c6127	USER	(NULL)
4	fulanbinfulan	7c232f58f03f6d00c8d28166fbff7a8b	USER	(NULL)
5	mawar	bd117502364227fd8c09098d31e11313	USER	(NULL)
6	melati	27e80ebc907bd57004986be9e6f2dd83	USER	(NULL)
7	dahlia	ee11cbb19052e40b07aac0ca060c23ee	USER	(NULL)
8	lily	89f288757f4d0693c99b007855fc075e	USER	(NULL)

Gambar 7 Hasil Tampil Pengguna

sedangkan berikut tampilan data pengguna hasil pemanggilan serupa dengan eksekusi login

1 SELECT * FROM pengguna WHERE username = 'admin' AND password = MD5('admin');

id	username	password	peran	login_terakhir
1	admin	21232f297a57a5a743894a0e4a801fc3	ADMIN	(NULL)

Gambar 8 Hasil Tampil Login

```
SELECT * FROM pengguna WHERE username = 'admin' AND password = MD5('admin');
```

Terjemahan:


Pilih (SELECT) semua field (*) dari tabel pengguna (FROM pengguna) dimana username-nya adalah admin (WHERE username = 'admin') dan passwordnya adalah MD5(admin) (AND password = MD5('admin'));

Contoh lain menggunakan function TIMESTAMPDIFF() yang digunakan untuk menghitung selisih waktu, dan CURDATE() yang digunakan untuk mengambil tanggal hari ini. Perintah berikut menggunakan fungsi tersebut untuk menghitung masa kerja karyawan dalam satuan tahun dengan menghitung selisih tahun antara tanggal masuk dengan tanggal hari ini.

```

1 SELECT id, nama_lengkap, tanggal_masuk, TIMESTAMPDIFF(YEAR, tanggal_masuk,
2 CURDATE()) masa_kerja_tahun FROM karyawan;
3

```



karyawan (8r x 4c)

id	nama_lengkap	tanggal_masuk	masa_kerja_tahun
1	Admin	2011-01-01	10
2	Tes User	2012-02-02	9
3	John Doe	2013-03-03	8
4	Fulan Bin Fulan	2014-04-04	7
5	Mawar Kurniani	2015-05-05	6
6	Melati Rahmawati	2016-06-06	5
7	Dahlia Setiani	2017-07-07	4
8	Lily Handayani	2018-08-08	3

Gambar 9 Hasil Tampil Masa Kerja Karyawan

```
SELECT id, nama_lengkap, tanggal_masuk, TIMESTAMPDIFF(YEAR, tanggal_masuk, CURDATE()) masa_kerja_tahun FROM karyawan;
```

Terjemahan:



Pilih (SELECT) field (id, nama_lengkap, tanggal_masuk,) hitung selisih tahun dan tampilkan sebagai field masa kerja tahun (TIMESTAMPDIFF(YEAR, K.tanggal_masuk, CURDATE()) masa_kerja_tahun) dari tabel karyawan (FROM karyawan).

Perintah di atas tidak seperti perintah select sebelumnya yang menampilkan semua field, tapi menampilkan field tertentu saja. Pada pembahasan berikutnya terdapat field buatan atau biasa disebut alias yaitu masa kerja tahun yang tidak terdapat pada field asli tetapi merupakan hasil perhitungan selisih tahun dan ditampilkan ke dalam tabel hasil query

Select & Count

Perintah COUNT bisa digunakan bersama SELECT untuk menampilkan kuantitas dari data pada tabel. Perhatikan untuk kuantitas berbeda dengan jumlah yang menggunakan perintah SUM.

Misalkan untuk melihat kuantitas dari tabel lokasi

```
SELECT COUNT(*) FROM lokasi;
```

perintah ini akan menampilkan kuantitas dari seluruh data lokasi, jika diperlukan alias untuk penamaan bisa menggunakan perintah

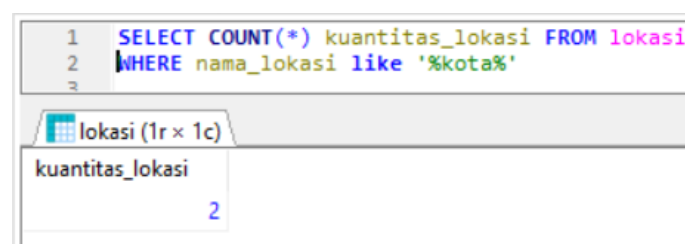
```
SELECT COUNT(*) AS kuantitas_lokasi FROM lokasi;
```

atau

```
SELECT COUNT(*) kuantitas_lokasi FROM lokasi;
```

Perintah tersebut bisa dikombinasikan dengan WHERE seperti pada contoh berikut

```
SELECT COUNT(*) kuantitas_lokasi FROM lokasi WHERE  
nama_lokasi like '%kota%'
```



Gambar 10 Hasil Tampil Kuantitas Lokasi dengan Filter

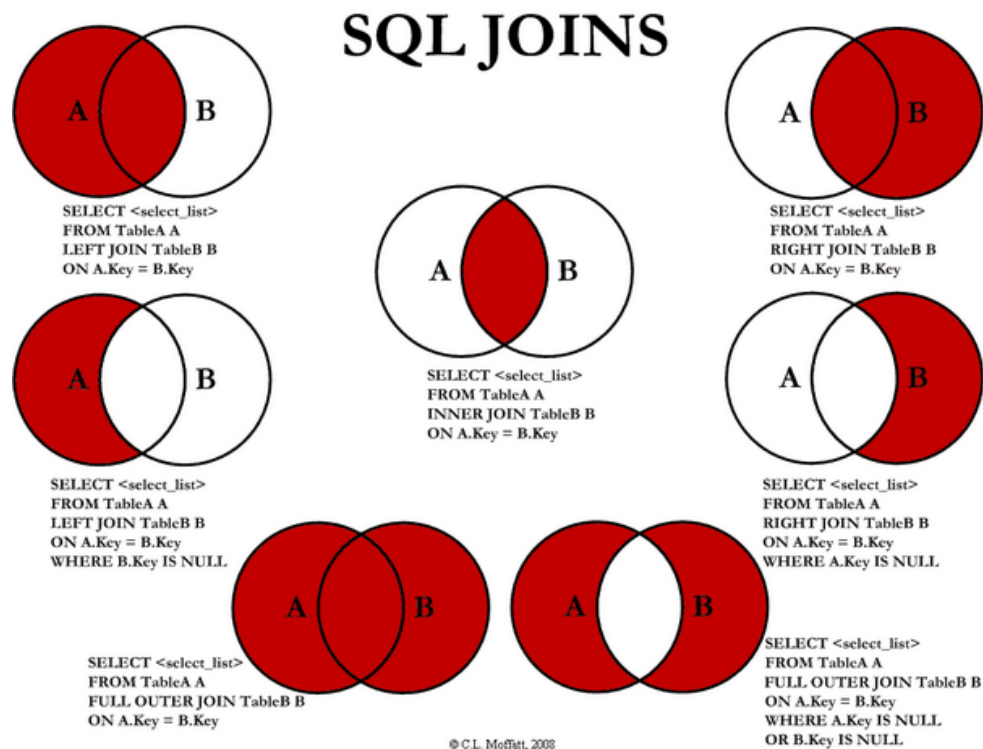
Latihan 2.3

Buat perintah SQL yang menampilkan kuantitas_jabatan yang nama_jabatan-nya mengandung kata `programmer`

kuantitas_jabatan
2

PERTEMUAN 3 JOIN

Perintah JOIN bisa digunakan bersama SELECT untuk penggabungan dua tabel atau lebih. JOIN memiliki beberapa jenis yaitu LEFT JOIN, INNER JOIN, RIGHT JOIN, dan FULL OUTER JOIN, yang dapat divisualisasikan pada gambar berikut.



diambil dari: <https://www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins>

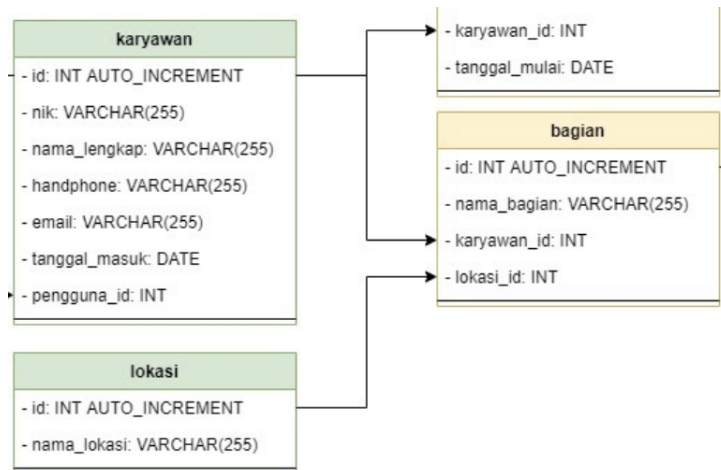
Penjelasan lebih lanjut bisa dilihat di:

<https://stackoverflow.com/a/38578>

OneTo Many

Pada modul ini terdapat beberapa tabel yang memiliki foreign key yang bisa dilakukan operasi JOIN di dalamnya, contoh pada tabel bagian





Gambar 11 Contoh One To Many

Tabel bagian memiliki 2 (dua) foreign key yaitu karyawan_id dan lokasi_id yang menunjukkan ada hubungan kepada 2 (dua) table lain. Jenis relasi pada ketiga tabel ini adalah ONE TO MANY. Jika untuk menampilkan hanya isi dari tabel bagian. Bisa digunakan perintah berikut

```
SELECT * FROM bagian
```

1 SELECT * FROM bagian			
bagian (3r x 4c)			
id	nama_bagian	karyawan_id	lokasi_id
1	Autentikasi	5	1
2	Data Science	3	1
3	Backend Developer	6	2

Gambar 12 Hasil Tampil Bagian

namun tentu tabel yang ditampilkan kurang informatif karena hanya menampilkan karyawan_id dan lokasi_id saja tanpa menampilkan nama lokasi maupun nama karyawannya. Sehingga dengan menerapkan INNER JOIN dan alias, perintah diatas bisa diubah menjadi

```
SELECT B.*, L.nama_lokasi lokasi_bagian FROM bagian B INNER
JOIN lokasi L ON B.lokasi_id = L.id
```

Terjemahan:

Pilih (SELECT) semua field pada tabel B (B.*,) field nama_lokasi dari tabel L (L.nama_lokasi) tampilkan dengan alias lokasi_bagian (lokasi_bagian) dari tabel bagian dengan alias tabel B (FROM bagian B) INNER JOIN kepada tabel lokasi dengan alias tabel L (lokasi L) dimana lokasi_id pada tabel B sama dengan id pada tabel L (ON B.lokasi_id = L.id)

1	SELECT B.*, L.nama_lokasi lokasi_bagian FROM bagian B
2	INNER JOIN lokasi L ON B.lokasi_id = L.id
3	

bagian (3r x 5c)					
id	nama_bagian	karyawan_id	lokasi_id	lokasi_bagian	
1	Autentikasi	5	1	Kota Banjarmasin	
2	Data Science	3	1	Kota Banjarmasin	
3	Backend Developer	6	2	Kota Banjarbaru	

Gambar 13 Hasil Tampil Lokasi Bagian

Latihan 3.1

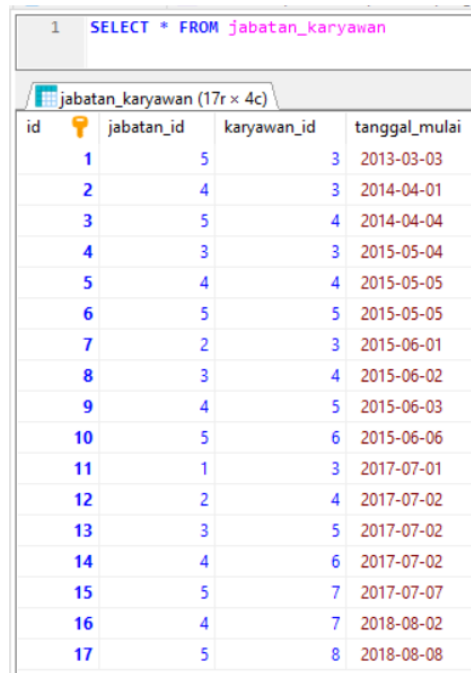
Buat perintah SQL yang menampilkan nama_kepala_bagian yang diambil dari nama_lengkap dari tabel karyawan yang terhubung dengan tabel bagian

id	nama_bagian	karyawan_id	lokasi_id	nama_kepala_bagian	lokasi_bagian
1	Autentikasi	5	1	Mawar Kurniani	Kota Banjarmasin
2	Data Science	3	1	John Doe	Kota Banjarmasin
3	Backend Developer	6	2	Melati Rahmawati	Kota Banjarbaru

Many To Many

Pada pembahasan sebelumnya sudah ditampilkan contoh relasi ONE TO MANY dimana satu bagian hanya memiliki satu lokasi, sedangkan satu lokasi bisa ada di beberapa bagian. Pada modul ini terdapat tabel yang memiliki relasi MANY TO MANY, contohnya adalah antara tabel jabatan dan tabel karyawan. Dimana satu jabatan bisa dijabat oleh

beberapa karyawan, dan satu karyawan bisa menjabat beberapa jabatan, dalam hal ini yang dimaksud adalah bukan rangkap jabatan melainkan merujuk kepada riwayat jabatan sang karyawan. Hubungan antara kedua tabel dituangkan ke dalam sebuah tabel yaitu tabel jabatan_karyawan. Berikut tampilan tabel tersebut.



id	jabatan_id	karyawan_id	tanggal_mulai
1	5	3	2013-03-03
2	4	3	2014-04-01
3	5	4	2014-04-04
4	3	3	2015-05-04
5	4	4	2015-05-05
6	5	5	2015-05-05
7	2	3	2015-06-01
8	3	4	2015-06-02
9	4	5	2015-06-03
10	5	6	2015-06-06
11	1	3	2017-07-01
12	2	4	2017-07-02
13	3	5	2017-07-02
14	4	6	2017-07-02
15	5	7	2017-07-07
16	4	7	2018-08-02
17	5	8	2018-08-08

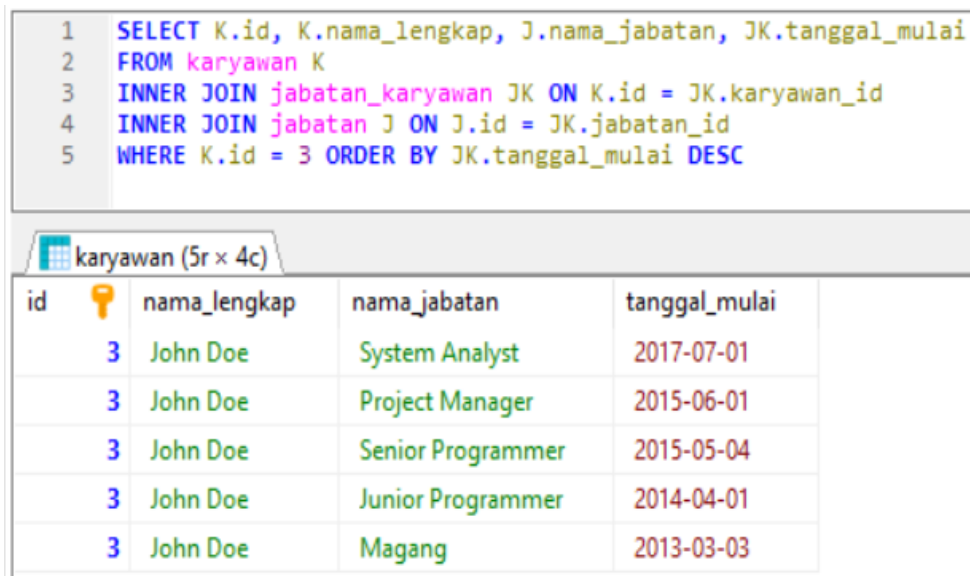
Gambar 14 Hasil Tampil Jabatan Karyawan

Latihan 3.2 Buat perintah SQL yang menampilkan nama_lengkap yang diambil dari tabel karyawan dan nama_jabatan dari tabel jabatan

id	jabatan_id	karyawan_id	tanggal_mulai	nama_jabatan	nama_lengkap
1	5	3	2013-03-03	Magang	John Doe
2	4	3	2014-04-01	Junior Programmer	John Doe
3	5	4	2014-04-04	Magang	Fulan Bin Fulan
4	3	3	2015-05-04	Senior Programmer	John Doe
5	4	4	2015-05-05	Junior Programmer	Fulan Bin Fulan
6	5	5	2015-05-05	Magang	Mawar Kurniani
7	2	3	2015-06-01	Project Manager	John Doe
8	3	4	2015-06-02	Senior Programmer	Fulan Bin Fulan
9	4	5	2015-06-03	Junior Programmer	Mawar Kurniani
10	5	6	2015-06-06	Magang	Melati Rahmawati

Terdapat beberapa cara untuk menampilkan data tersebut menjadi lebih informatif. Cara pertama adalah menampilkan data riwayat jabatan per karyawan. Misalkan pada karyawan dengan nama John Doe yang memiliki id=3, data akan ditampilkanurut berdasarkan tanggal_mulai pada tabel jabatan_karyawan dari yang terbesar

```
SELECT K.id, K.nama_lengkap, J.nama_jabatan,  
JK.tanggal_mulai  
FROM karyawan K  
INNER JOIN jabatan_karyawan JK ON K.id = JK.karyawan_id  
INNER JOIN jabatan J ON J.id = JK.jabatan_id  
WHERE K.id = 3 ORDER BY JK.tanggal_mulai DESC
```



```
1 SELECT K.id, K.nama_lengkap, J.nama_jabatan, JK.tanggal_mulai
2 FROM karyawan K
3 INNER JOIN jabatan_karyawan JK ON K.id = JK.karyawan_id
4 INNER JOIN jabatan J ON J.id = JK.jabatan_id
5 WHERE K.id = 3 ORDER BY JK.tanggal_mulai DESC
```

id	nama_lengkap	nama_jabatan	tanggal_mulai
3	John Doe	System Analyst	2017-07-01
3	John Doe	Project Manager	2015-06-01
3	John Doe	Senior Programmer	2015-05-04
3	John Doe	Junior Programmer	2014-04-01
3	John Doe	Magang	2013-03-03

Gambar 15 Hasil Tampil Jabatan

Cara berikutnya adalah menampilkan data karyawan beserta jabatannya. Data yang ditampilkan berhasil menampilkan karyawan beserta jabatannya, karena sifat relasinya yang MANY TO MANY maka seluruh riwayat jabatan pun ikut ditampilkan.

1	SELECT K.id, K.nama_lengkap, J.nama_jabatan, JK.tanggal_mulai
2	FROM karyawan K
3	INNER JOIN jabatan_karyawan JK ON K.id = JK.karyawan_id
4	INNER JOIN jabatan J ON J.id = JK.jabatan_id
5	ORDER BY K.id

karyawan (17r x 4c)			
id	nama_lengkap	nama_jabatan	tanggal_mulai
3	John Doe	System Analyst	2017-07-01
3	John Doe	Magang	2013-03-03
3	John Doe	Junior Programmer	2014-04-01
3	John Doe	Senior Programmer	2015-05-04
3	John Doe	Project Manager	2015-06-01
4	Fulan Bin Fulan	Project Manager	2017-07-02
4	Fulan Bin Fulan	Magang	2014-04-04
4	Fulan Bin Fulan	Junior Programmer	2015-05-05
4	Fulan Bin Fulan	Senior Programmer	2015-06-02
5	Mawar Kurniani	Senior Programmer	2017-07-02
5	Mawar Kurniani	Magang	2015-05-05
5	Mawar Kurniani	Junior Programmer	2015-06-03
6	Melati Rahmawati	Junior Programmer	2017-07-02
6	Melati Rahmawati	Magang	2015-06-06
7	Dahlia Setiani	Magang	2017-07-07
7	Dahlia Setiani	Junior Programmer	2018-08-02
8	Lily Handayani	Magang	2018-08-08

Gambar 16 Hasil Tampil Seluruh Jabatan

Jika yang diperlukan hanya jabatan terakhir saja maka perintah yang digunakan untuk menampilkannya adalah

```
SELECT K.id, K.nama_lengkap,
(
    SELECT J.nama_jabatan FROM jabatan_karyawan JK
    INNER JOIN jabatan J ON JK.jabatan_id = J.id
    WHERE karyawan_id = K.id ORDER BY JK.tanggal_mulai DESC
LIMIT 1
) jabatan_terakhir
FROM karyawan K
```

```

1 SELECT K.id, K.nama_lengkap,
2 (
3     SELECT J.nama_jabatan FROM jabatan_karyawan JK
4     INNER JOIN jabatan J ON JK.jabatan_id = J.id
5     WHERE karyawan_id = K.id ORDER BY JK.tanggal_mulai DESC LIMIT 1
6 ) jabatan_terakhir
7 FROM karyawan K

```

id	nama_lengkap	jabatan_terakhir
1	Admin	(NULL)
2	Tes User	(NULL)
3	John Doe	System Analyst
4	Fulan Bin Fulan	Project Manager
5	Mawar Kurniani	Senior Programmer
6	Melati Rahmawati	Junior Programmer
7	Dahlia Setiani	Junior Programmer
8	Lily Handayani	Magang

Pada perintah tersebut terdapat nesting SELECT yang pada kasus ini berfungsi untuk mengambil nama_jabatan (SELECT J.nama_jabatan) terakhir ditandai dengan perintah ORDER BY JK.tanggal_mulai DESC dan 1 (satu) jabatan, ditandai dengan perintah LIMIT 1. perlu diperhatikan bahwa untuk nesting SELECT harus memiliki 1 output saja, oleh karenanya field yang diambil hanya nama_jabatan saja, dan diberikan batasan hanya 1 baris data saja (LIMIT 1)

Latihan 3.3 Buat perintah SQL yang menampilkan data karyawan beserta jabatan terakhirnya, dan tanggal mulai menjabat jabatan terakhir tersebut

id	nama_lengkap	jabatan_terakhir	tanggal_mulai_terakhir
1	Admin	(NULL)	(NULL)
2	Tes User	(NULL)	(NULL)
3	John Doe	System Analyst	2017-07-01
4	Fulan Bin Fulan	Project Manager	2017-07-02
5	Mawar Kurniani	Senior Programmer	2017-07-02
6	Melati Rahmawati	Junior Programmer	2017-07-02
7	Dahlia Setiani	Junior Programmer	2018-08-02
8	Lily Handayani	Magang	2018-08-08

Cara lain menyajikan informasi jabatan_karyawan adalah dengan menampilkan seluruh jabatan beserta jumlah karyawan yang menjabatnya

```
SELECT
```

```
(
```

```
    SELECT J.nama_jabatan FROM jabatan_karyawan JK
```

```
    INNER JOIN jabatan J ON JK.jabatan_id = J.id
```

```
    WHERE karyawan_id = K.id
```

```
    ORDER BY JK.tanggal_mulai DESC LIMIT 1
```

```
) jabatan_terakhir,
```

```
COUNT(*) jumlah
```

```
FROM karyawan K
```

```
GROUP BY jabatan_terakhir
```

```

1 SELECT
2 (
3     SELECT J.nama_jabatan FROM jabatan_karyawan JK
4     INNER JOIN jabatan J ON JK.jabatan_id = J.id
5     WHERE karyawan_id = K.id
6     ORDER BY JK.tanggal_mulai DESC LIMIT 1
7 ) jabatan_terakhir,
8 COUNT(*) jumlah
9 FROM karyawan K
10 GROUP BY jabatan_terakhir

```

jabatan_terakhir	jumlah
(NULL)	2
Junior Programmer	2
Magang	1
Project Manager	1
Senior Programmer	1
System Analyst	1

Gambar 17 Hasil Tampil Jumlah Karyawan pada Jabatan

Latihan 3.4 Buat Perintah SQL yang menyajikan data karyawan beserta bagian terakhirnya.

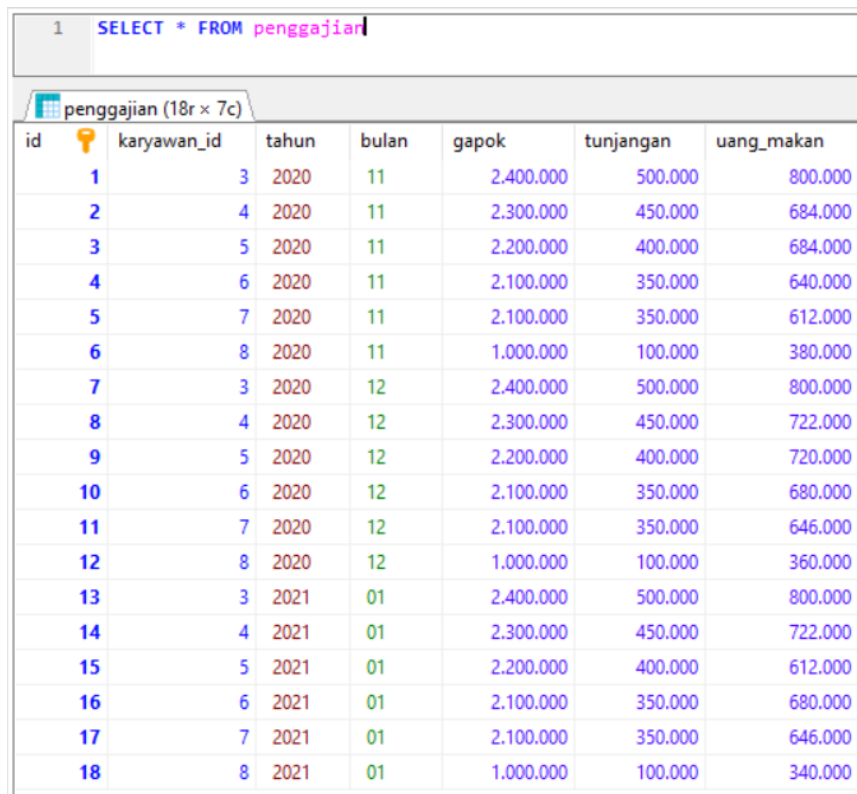
id	nama_lengkap	jabatan_terakhir	tanggal_jabatan_terakhir	bagian_terakhir	tanggal_bagian_terakhir
1	Admin	(NULL)	(NULL)	(NULL)	(NULL)
2	Tes User	(NULL)	(NULL)	(NULL)	(NULL)
3	John Doe	System Analyst	2017-07-01	Data Science	2018-04-01
4	Fulan Bin Fulan	Project Manager	2017-07-02	Autentikasi	2018-03-03
5	Mawar Kurniani	Senior Programmer	2017-07-02	Autentikasi	2018-03-03
6	Melati Rahmawati	Junior Programmer	2017-07-02	Backend Developer	2018-04-04
7	Dahlia Setiani	Junior Programmer	2018-08-02	Data Science	2018-04-01
8	Lily Handayani	Magang	2018-08-08	Backend Developer	2018-04-04

bagian_terakhir	jumlah
(NULL)	2
Autentikasi	2
Backend Developer	2
Data Science	2

PERTEMUAN 4 PENGGAJIAN & PRESENSI

Penggajian

Pada tabel penggajian yang sudah diisi pada Pertemuan 1, jika dilakukan SELECT pada datanya maka tampil seperti gambar berikut



1 SELECT * FROM penggajian						
penggajian (18r x 7c)						
id	karyawan_id	tahun	bulan	gapok	tunjangan	uang_makan
1	3	2020	11	2.400.000	500.000	800.000
2	4	2020	11	2.300.000	450.000	684.000
3	5	2020	11	2.200.000	400.000	684.000
4	6	2020	11	2.100.000	350.000	640.000
5	7	2020	11	2.100.000	350.000	612.000
6	8	2020	11	1.000.000	100.000	380.000
7	3	2020	12	2.400.000	500.000	800.000
8	4	2020	12	2.300.000	450.000	722.000
9	5	2020	12	2.200.000	400.000	720.000
10	6	2020	12	2.100.000	350.000	680.000
11	7	2020	12	2.100.000	350.000	646.000
12	8	2020	12	1.000.000	100.000	360.000
13	3	2021	01	2.400.000	500.000	800.000
14	4	2021	01	2.300.000	450.000	722.000
15	5	2021	01	2.200.000	400.000	612.000
16	6	2021	01	2.100.000	350.000	680.000
17	7	2021	01	2.100.000	350.000	646.000
18	8	2021	01	1.000.000	100.000	340.000

Terdapat beberapa cara menyajikan data pada tabel tersebut agar menjadi lebih informatif.

Jumlah Gaji yang Dibayarkan Per Tahun

```
SELECT tahun,  
       SUM(P.gapok) +  
       SUM(P.tunjangan) +  
       SUM(P.uang_makan)  
       jumlah_bayar_gaji  
FROM penggajian P  
GROUP BY tahun;
```

1	SELECT tahun,
2	SUM(P.gapok) +
3	SUM(P.tunjangan) +
4	SUM(P.uang_makan)
5	jumlah_bayar_gaji
6	FROM penggajian P
7	GROUP BY tahun;

penggajian (2r × 2c)	
tahun	jumlah_bayar_gaji
2020	36.228.000
2021	18.050.000

Gambar 18 Hasil Tampil Jumlah Bayar Gaji

Rincian Jumlah Gaji yang Dibayarkan Per Tahun

```
SELECT tahun,
       SUM(P.gapok) jumlah_gapok,
       SUM(P.tunjangan) jumlah_tunjangan,
       SUM(P.uang_makan) jumlah_uang_makan
FROM penggajian P
GROUP BY tahun;
```

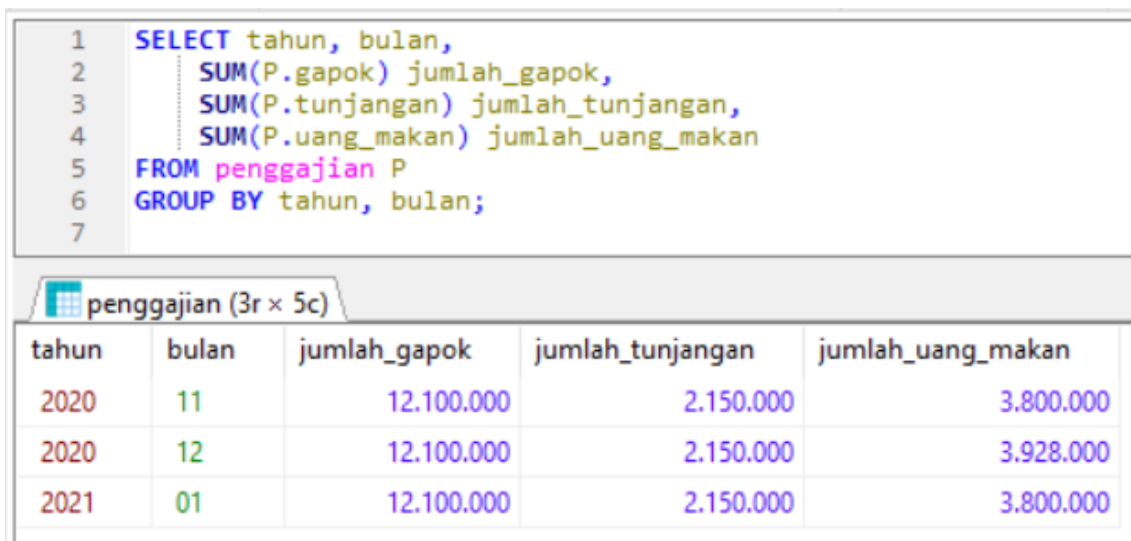
1	SELECT tahun,
2	SUM(P.gapok) jumlah_gapok,
3	SUM(P.tunjangan) jumlah_tunjangan,
4	SUM(P.uang_makan) jumlah_uang_makan
5	FROM penggajian P
6	GROUP BY tahun;

penggajian (2r × 4c)			
tahun	jumlah_gapok	jumlah_tunjangan	jumlah_uang_makan
2020	24.200.000	4.300.000	7.728.000
2021	12.100.000	2.150.000	3.800.000

Gambar 19 Hasil Tampil Rincian Jumlah Gaji yang Dibayarkan Per Tahun

Rincian Jumlah Gaji yang Dibayarkan Per Bulan

```
SELECT tahun, bulan,  
       SUM(P.gapok) jumlah_gapok,  
       SUM(P.tunjangan) jumlah_tunjangan,  
       SUM(P.uang_makan) jumlah_uang_makan  
FROM penggajian P  
GROUP BY tahun, bulan;
```



The screenshot shows a SQL query editor with the following code:

```
1 SELECT tahun, bulan,  
2     SUM(P.gapok) jumlah_gapok,  
3     SUM(P.tunjangan) jumlah_tunjangan,  
4     SUM(P.uang_makan) jumlah_uang_makan  
5 FROM penggajian P  
6 GROUP BY tahun, bulan;  
7
```

Below the query, a table titled "penggajian (3r x 5c)" displays the results:

tahun	bulan	jumlah_gapok	jumlah_tunjangan	jumlah_uang_makan
2020	11	12.100.000	2.150.000	3.800.000
2020	12	12.100.000	2.150.000	3.928.000
2021	01	12.100.000	2.150.000	3.800.000

Gambar 20 Hasil Tampil Rincian Jumlah Gaji yang Dibayarkan Per Bulan

Rincian Jumlah Gaji yang Dibayarkan Per Karyawan dalam 1 Tahun

```
SELECT P.tahun,  
       P.karyawan_id,  
       K.nama_lengkap,  
       SUM(P.gapok) jumlah_gapok,  
       SUM(P.tunjangan) jumlah_tunjangan,  
       SUM(P.uang_makan) jumlah_uang_makan  
FROM penggajian P  
LEFT JOIN karyawan K ON P.karyawan_id = K.id  
WHERE P.tahun = "2020"  
GROUP BY P.karyawan_id
```

```

1 SELECT P.tahun,
2       P.karyawan_id,
3       K.nama_lengkap,
4       SUM(P.gapok) jumlah_gapok,
5       SUM(P.tunjangan) jumlah_tunjangan,
6       SUM(P.uang_makan) jumlah_uang_makan
7 FROM penggajian P
8 LEFT JOIN karyawan K ON P.karyawan_id = K.id
9 WHERE P.tahun = "2020"
10 GROUP BY P.karyawan_id

```

tahun	karyawan_id	nama_lengkap	jumlah_gapok	jumlah_tunjangan	jumlah_uang_makan
2020	3	John Doe	4.800.000	1.000.000	1.600.000
2020	4	Fulan Bin Fulan	4.600.000	900.000	1.406.000
2020	5	Mawar Kurniani	4.400.000	800.000	1.404.000
2020	6	Melati Rahmawati	4.200.000	700.000	1.320.000
2020	7	Dahlia Setiani	4.200.000	700.000	1.258.000
2020	8	Lily Handayani	2.000.000	200.000	740.000

Gambar 21 Hasil Tampil Rincian Jumlah Gaji yang Dibayarkan Per Karyawan dalam 1 Tahun

Rincian Bulanan Jumlah Gaji yang Dibayarkan 1 Karyawan dalam 1 Tahun

```

SELECT P.tahun, P.bulan,
       P.karyawan_id,
       K.nama_lengkap,
       P.gapok,
       P.tunjangan,
       P.uang_makan
FROM penggajian P
LEFT JOIN karyawan K ON P.karyawan_id = K.id
WHERE P.tahun = "2020" AND karyawan_id = 3

```



```

1 SELECT P.tahun, P.bulan,
2       P.karyawan_id,
3       K.nama_lengkap,
4       P.gapok,
5       P.tunjangan,
6       P.uang_makan
7 FROM penggajian P
8 LEFT JOIN karyawan K ON P.karyawan_id = K.id
9 WHERE P.tahun = "2020" AND karyawan_id = 3

```

tahun	bulan	karyawan_id	nama_lengkap	gapok	tunjangan	uang_makan
2020	11	3	John Doe	2.400.000	500.000	800.000
2020	12	3	John Doe	2.400.000	500.000	800.000

Gambar 22 Rincian Bulanan Jumlah Gaji yang Dibayarkan 1 Karyawan dalam 1 Tahun

Slip Gaji

```

SELECT P.tahun, P.bulan,
       P.karyawan_id,
       K.nama_lengkap,
       P.gapok,
       P.tunjangan,
       P.uang_makan
FROM penggajian P
LEFT JOIN karyawan K ON P.karyawan_id = K.id
WHERE P.tahun = "2020" AND P.bulan = "12" AND karyawan_id =
3

```

```

1 SELECT P.tahun, P.bulan,
2       P.karyawan_id,
3       K.nama_lengkap,
4       P.gapok,
5       P.tunjangan,
6       P.uang_makan
7 FROM penggajian P
8 LEFT JOIN karyawan K ON P.karyawan_id = K.id
9 WHERE P.tahun = "2020" AND P.bulan = "12" AND karyawan_id = 3

```

tahun	bulan	karyawan_id	nama_lengkap	gapok	tunjangan	uang_makan
2020	12	3	John Doe	2.400.000	500.000	800.000

Gambar 23 Hasil Tampil Slip Gaji



Presensi

Tabel presensi jika ditampilkan datanya maka hasilnya sebagai berikut

1 `SELECT * FROM presensi`

presensi (552r x 6c)						
id	karyawan_id	tanggal	jam_masuk	jam_keluar	keterangan	
1	3	2020-11-01	(NULL)	(NULL)	AKHIR PEKAN	
2	3	2020-11-02	08:00:00	16:00:00	HADIR	
3	3	2020-11-03	08:00:00	16:00:00	HADIR	
4	3	2020-11-04	08:00:00	16:00:00	HADIR	
5	3	2020-11-05	08:00:00	16:00:00	HADIR	
6	3	2020-11-06	08:00:00	16:00:00	HADIR	
7	3	2020-11-07	(NULL)	(NULL)	AKHIR PEKAN	
8	3	2020-11-08	(NULL)	(NULL)	AKHIR PEKAN	
9	3	2020-11-09	08:00:00	16:00:00	HADIR	
10	3	2020-11-10	(NULL)	(NULL)	LIBUR NASIONAL	

Gambar 24 Hasil Tampil Presensi

Data yang tampil belum informatif bagi pengguna karena banyak menampilkan id dan terlalu rinci. Sehingga perlu disajikan dalam bentuk yang lebih baik

Jumlah Presensi Seluruh Karyawan per Keterangan Setiap Tahun

Perintah berikut menampilkan jumlah presensi seluruh karyawan dengan dibagi berdasarkan keterangan presensinya. Informasi seperti ini membantu pengguna untuk melakukan evaluasi terhadap tren tingkat kehadiran seluruh karyawan per tahunnya.

```
SELECT YEAR(tanggal) tahun,  
SUM(case when keterangan = 'HADIR' then 1 else 0 end)  
jumlah_hadir,  
SUM(case when keterangan = 'SAKIT' then 1 else 0 end)  
jumlah_sakit,  
SUM(case when keterangan = 'IZIN' then 1 else 0 end)  
jumlah_izin,
```



```

SUM(case when keterangan = 'CUTI' then 1 else 0 end)
jumlah_cuti,
SUM(case when keterangan = 'AKHIR PEKAN' then 1 else 0 end)
jumlah_akhir_pekan,
SUM(case when keterangan = 'LIBUR NASIONAL' then 1 else 0
end) jumlah_libur_nasional,
SUM(case when keterangan = 'TANPA KETERANGAN' then 1 else 0
end) jumlah_tanpa_keterangan,
COUNT(*) total
FROM `presensi` GROUP BY tahun

```

```

1 SELECT YEAR(tanggal) tahun,
2 SUM(case when keterangan = 'HADIR' then 1 else 0 end) jumlah_hadir,
3 SUM(case when keterangan = 'SAKIT' then 1 else 0 end) jumlah_sakit,
4 SUM(case when keterangan = 'IZIN' then 1 else 0 end) jumlah_izin,
5 SUM(case when keterangan = 'CUTI' then 1 else 0 end) jumlah_cuti,
6 SUM(case when keterangan = 'AKHIR PEKAN' then 1 else 0 end) jumlah_akhir_pekan,
7 SUM(case when keterangan = 'LIBUR NASIONAL' then 1 else 0 end) jumlah_libur_nasional,
8 SUM(case when keterangan = 'TANPA KETERANGAN' then 1 else 0 end) jumlah_tanpa_keterangan,
9 COUNT(*) total
10 FROM `presensi` GROUP BY tahun

```

tahun	jumlah_hadir	jumlah_sakit	jumlah_izin	jumlah_cuti	jumlah_akhir_pekan	jumlah_libur_nasional	jumlah_tanpa_keterangan	total
2.020	230	5	3	2	102	24	0	366
2.021	112	4	4	0	60	6	0	186

Gambar 25 Jumlah Presensi Seluruh Karyawan per Keterangan Setiap Tahun

Perintah ini juga bisa dikombinasikan dengan WHERE tahun tertentu jika ingin menampilkan berdasarkan tahunnya.

```

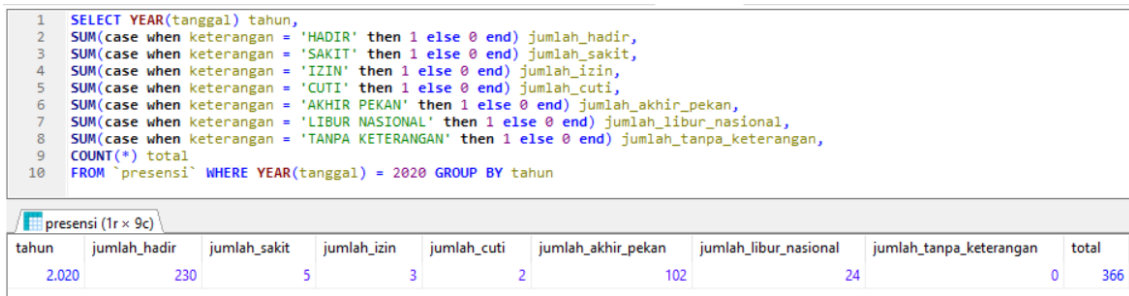
SELECT YEAR(tanggal) tahun,
SUM(case when keterangan = 'HADIR' then 1 else 0 end)
jumlah_hadir,
SUM(case when keterangan = 'SAKIT' then 1 else 0 end)
jumlah_sakit,
SUM(case when keterangan = 'IZIN' then 1 else 0 end)
jumlah_izin,
SUM(case when keterangan = 'CUTI' then 1 else 0 end)
jumlah_cuti,
SUM(case when keterangan = 'AKHIR PEKAN' then 1 else 0 end)
jumlah_akhir_pekan,

```

```

SUM(case when keterangan = 'LIBUR NASIONAL' then 1 else 0
end) jumlah_libur_nasional,
SUM(case when keterangan = 'TANPA KETERANGAN' then 1 else 0
end) jumlah_tanpa_keterangan,
COUNT(*) total
FROM `presensi` WHERE YEAR(tanggal) = 2020 GROUP BY tahun

```



```

1 SELECT YEAR(tanggal) tahun,
2 SUM(case when keterangan = 'HADIR' then 1 else 0 end) jumlah_hadir,
3 SUM(case when keterangan = 'SAKIT' then 1 else 0 end) jumlah_sakit,
4 SUM(case when keterangan = 'IZIN' then 1 else 0 end) jumlah_izin,
5 SUM(case when keterangan = 'CUTI' then 1 else 0 end) jumlah_cuti,
6 SUM(case when keterangan = 'AKHIR PEKAN' then 1 else 0 end) jumlah_akhir_pekan,
7 SUM(case when keterangan = 'LIBUR NASIONAL' then 1 else 0 end) jumlah_libur_nasional,
8 SUM(case when keterangan = 'TANPA KETERANGAN' then 1 else 0 end) jumlah_tanpa_keterangan,
9 COUNT(*) total
10 FROM `presensi` WHERE YEAR(tanggal) = 2020 GROUP BY tahun

```

tahun	jumlah_hadir	jumlah_sakit	jumlah_izin	jumlah_cuti	jumlah_akhir_pekan	jumlah_libur_nasional	jumlah_tanpa_keterangan	total
2.020	230	5	3	2	102	24	0	366

Gambar 26 Jumlah Presensi Seluruh Karyawan per Keterangan Dalam 1 Tahun

Rincian Bulanan Jumlah Presensi Seluruh Karyawan per Keterangan

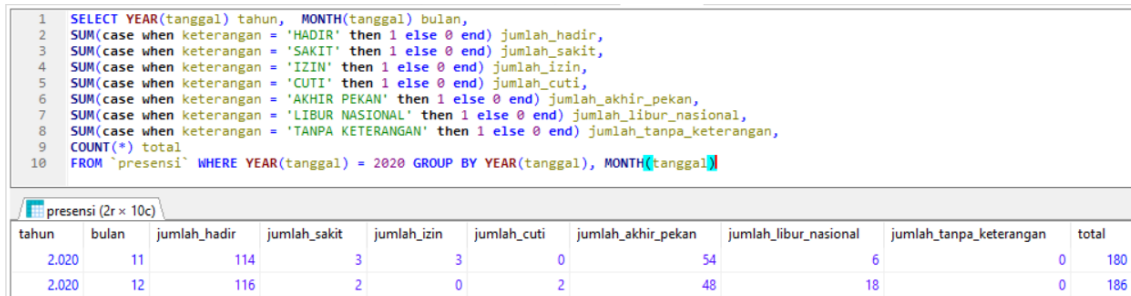
Pada perintah sebelumnya, data disajikan dalam bentuk jumlah tahunan. Data tersebut bisa dibuat lebih rinci dengan menampilkan rincian jumlah presensi setiap bulannya.

```

SELECT YEAR(tanggal) tahun, MONTH(tanggal) bulan,
SUM(case when keterangan = 'HADIR' then 1 else 0 end)
jumlah_hadir,
SUM(case when keterangan = 'SAKIT' then 1 else 0 end)
jumlah_sakit,
SUM(case when keterangan = 'IZIN' then 1 else 0 end)
jumlah_izin,
SUM(case when keterangan = 'CUTI' then 1 else 0 end)
jumlah_cuti,
SUM(case when keterangan = 'AKHIR PEKAN' then 1 else 0 end)
jumlah_akhir_pekan,
SUM(case when keterangan = 'LIBUR NASIONAL' then 1 else 0
end) jumlah_libur_nasional,
SUM(case when keterangan = 'TANPA KETERANGAN' then 1 else 0
end) jumlah_tanpa_keterangan,

```

```
COUNT(*) total
FROM `presensi` WHERE YEAR(tanggal) = 2020 GROUP BY tahun,
bulan
```



The screenshot shows a SQL query in a database client window. The query is a SELECT statement that groups presensi data by year and month, calculating various attendance metrics using SUM and CASE statements. Below the query, the results are displayed in a table with 10 columns: tahun, bulan, jumlah_hadir, jumlah_sakit, jumlah_izin, jumlah_cuti, jumlah_akhir_pekan, jumlah_libur_nasional, jumlah_tanpa_keterangan, and total. The results show data for the years 2020 and 2021, with months 11 and 12.

tahun	bulan	jumlah_hadir	jumlah_sakit	jumlah_izin	jumlah_cuti	jumlah_akhir_pekan	jumlah_libur_nasional	jumlah_tanpa_keterangan	total
2.020	11	114	3	3	0	54	6	0	180
2.020	12	116	2	0	2	48	18	0	186

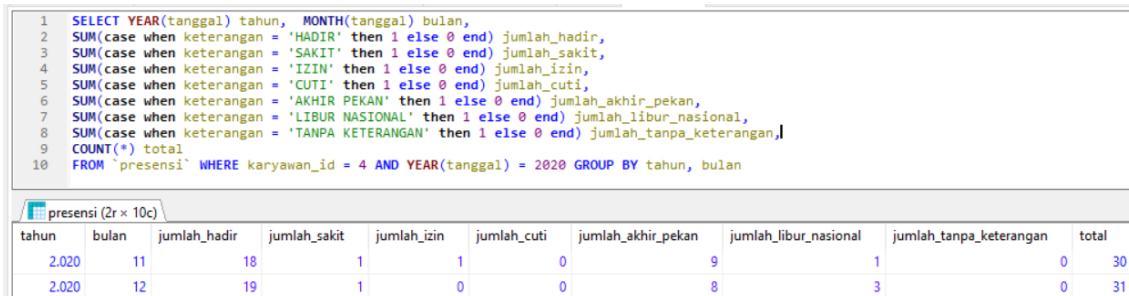
Gambar 27 Rincian Bulanan Jumlah Presensi Seluruh Karyawan per Keterangan

Rincian Bulanan Jumlah Presensi 1 Karyawan per Keterangan

Pada perintah sebelumnya, data disajikan dalam bentuk jumlah bulanan untuk seluruh karyawan. Data tersebut bisa dibuat lebih rinci dengan menampilkan rincian jumlah presensi untuk satu karyawan tertentu.

```
SELECT YEAR(tanggal) tahun, MONTH(tanggal) bulan,
SUM(case when keterangan = 'HADIR' then 1 else 0 end)
jumlah_hadir,
SUM(case when keterangan = 'SAKIT' then 1 else 0 end)
jumlah_sakit,
SUM(case when keterangan = 'IZIN' then 1 else 0 end)
jumlah_izin,
SUM(case when keterangan = 'CUTI' then 1 else 0 end)
jumlah_cuti,
SUM(case when keterangan = 'AKHIR PEKAN' then 1 else 0 end)
jumlah_akhir_pekan,
SUM(case when keterangan = 'LIBUR NASIONAL' then 1 else 0
end) jumlah_libur_nasional,
SUM(case when keterangan = 'TANPA KETERANGAN' then 1 else 0
end) jumlah_tanpa_keterangan,
```

```
COUNT(*) total
FROM `presensi` WHERE karyawan_id = 4 AND YEAR(tanggal) =
2020 GROUP BY tahun, bulan
```



```
1 SELECT YEAR(tanggal) tahun, MONTH(tanggal) bulan,
2 SUM(case when keterangan = 'HADIR' then 1 else 0 end) jumlah_hadir,
3 SUM(case when keterangan = 'SAKIT' then 1 else 0 end) jumlah_sakit,
4 SUM(case when keterangan = 'IZIN' then 1 else 0 end) jumlah_izin,
5 SUM(case when keterangan = 'CUTI' then 1 else 0 end) jumlah_cuti,
6 SUM(case when keterangan = 'AKHIR PEKAN' then 1 else 0 end) jumlah_akhir_pekan,
7 SUM(case when keterangan = 'LIBUR NASIONAL' then 1 else 0 end) jumlah_libur_nasional,
8 SUM(case when keterangan = 'TANPA KETERANGAN' then 1 else 0 end) jumlah_tanpa_keterangan,
9 COUNT(*) total
10 FROM `presensi` WHERE karyawan_id = 4 AND YEAR(tanggal) = 2020 GROUP BY tahun, bulan
```

tahun	bulan	jumlah_hadir	jumlah_sakit	jumlah_izin	jumlah_cuti	jumlah_akhir_pekan	jumlah_libur_nasional	jumlah_tanpa_keterangan	total
2.020	11	18	1	1	0	9	1	0	30
2.020	12	19	1	0	0	8	3	0	31

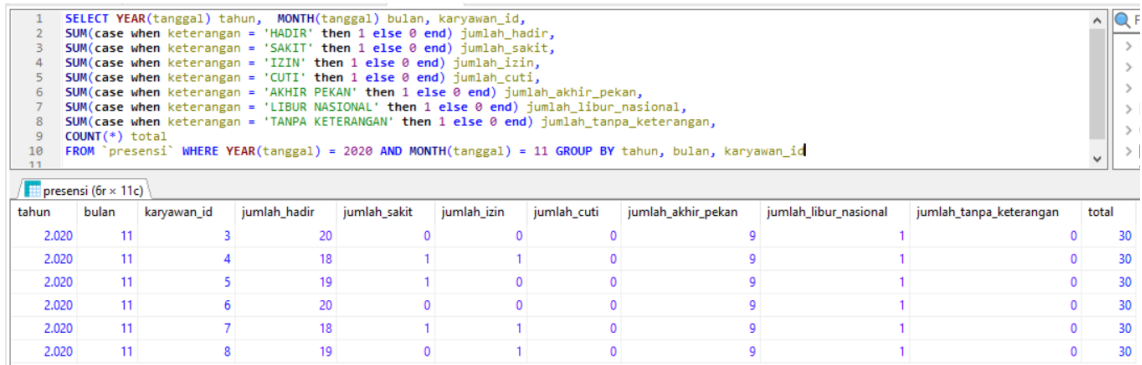
Gambar 28 Rincian Bulanan Jumlah Presensi 1 Karyawan per Keterangan

Rincian 1 Bulan Jumlah Presensi 1 Karyawan per Keterangan

Data juga bisa disajikan dengan menampilkan tingkat kehadiran tiap karyawan dalam 1 bulan dengan perintah sebagai berikut

```
SELECT YEAR(tanggal) tahun, MONTH(tanggal) bulan,
karyawan_id,
SUM(case when keterangan = 'HADIR' then 1 else 0 end)
jumlah_hadir,
SUM(case when keterangan = 'SAKIT' then 1 else 0 end)
jumlah_sakit,
SUM(case when keterangan = 'IZIN' then 1 else 0 end)
jumlah_izin,
SUM(case when keterangan = 'CUTI' then 1 else 0 end)
jumlah_cuti,
SUM(case when keterangan = 'AKHIR PEKAN' then 1 else 0 end)
jumlah_akhir_pekan,
SUM(case when keterangan = 'LIBUR NASIONAL' then 1 else 0
end) jumlah_libur_nasional,
SUM(case when keterangan = 'TANPA KETERANGAN' then 1 else 0
end) jumlah_tanpa_keterangan,
COUNT(*) total
```

```
FROM `presensi` WHERE YEAR(tanggal) = 2020 AND  
MONTH(tanggal) = 11 GROUP BY tahun, bulan, karyawan_id
```



The screenshot shows a SQL query in a database interface, followed by a table of results. The query is a SELECT statement that calculates various attendance metrics for each employee in November 2020. The results table has 11 columns: tahun, bulan, karyawan_id, jumlah_hadir, jumlah_sakit, jumlah_izin, jumlah_cuti, jumlah_akhir_pekan, jumlah_libur_nasional, jumlah_tanpa_keterangan, and total. There are 6 rows of data, one for each employee (karyawan_id 3 to 8).

tahun	bulan	karyawan_id	jumlah_hadir	jumlah_sakit	jumlah_izin	jumlah_cuti	jumlah_akhir_pekan	jumlah_libur_nasional	jumlah_tanpa_keterangan	total
2.020	11	3	20	0	0	0	9	1	0	30
2.020	11	4	18	1	1	0	9	1	0	30
2.020	11	5	19	1	0	0	9	1	0	30
2.020	11	6	20	0	0	0	9	1	0	30
2.020	11	7	18	1	1	0	9	1	0	30
2.020	11	8	19	0	1	0	9	1	0	30

Gambar 29 Rincian 1 Bulan Jumlah Presensi 1 Karyawan per Keterangan