SELECT

Brigida Arie Minartiningtyas, M.Kom.


Distinct

- ✓ Query ini digunakan untuk menghapus atau mengeleminasi duplikasi dari hasil tampilan SELECT

- ✓ Query **DISTINCT** digunakan **hanya untuk memanipulasi tampilan hasil dari tabel**. Duplikasi yang dihapus adalah untuk tampilan data, **BUKAN** tabel asli dari **MySQL**

NIP_dosen	nama_dosen	no_hp	alamat
0160436012	Sabrina Sari	0812349900	Pekanbaru
0173551078	Aria Sulistya	0880743523	Jakarta
0260432002	Maya Ari Putri	0812342342	Palembang
0275430005	Susi Indriani	0812656532	Bogor
0360432014	Suci Syuhada	0812341122	Palembang
0480432066	Tia Santrini	0812451177	Padang
0576431001	M. Siddiq	0812979005	Jakarta
0770435006	Rubin Hadi	0812567678	Papua
0785531001	Siswanto	0852878006	Padang
0867221006	Rudi Arwana	0823987598	Jakarta
0869437003	Mustalifah	0812338877	Aceh
1080432007	Arif Budiman	0812456345	Makasar

Yang terdaftar di table dosen berasal dari kota mana saja? Tampilkan alamat dari daftar_dosen dan diurutkan berdasarkan alamat

```
SELECT alamat
FROM daftar_dosen
ORDER BY alamat;
```

```
SELECT DISTINCT nama_kolom
FROM nama_tabel;
```

```
SELECT DISTINCT alamat
FROM daftar_dosen
ORDER BY alamat;
```

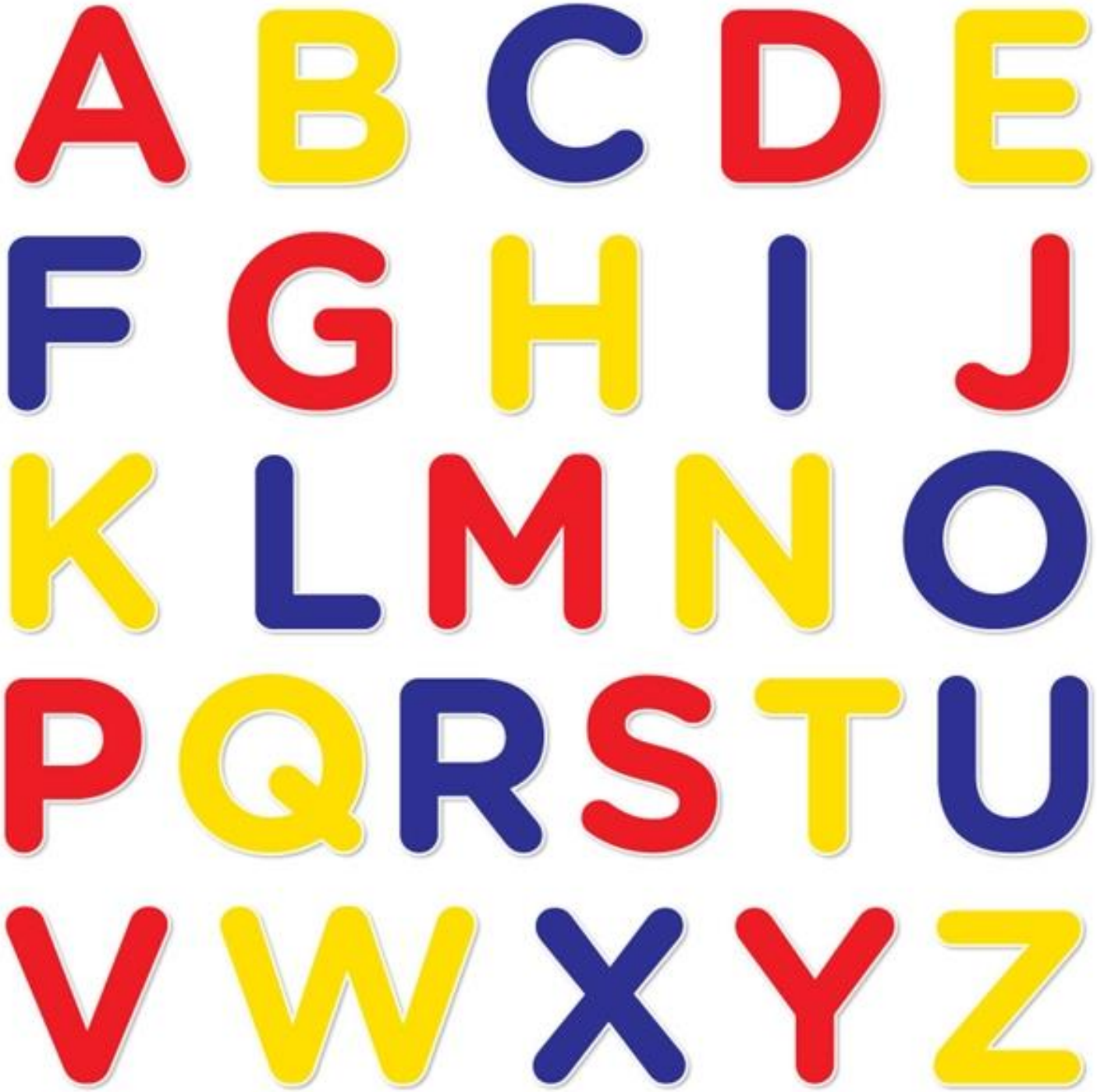
- ✓ Dengan penambahan perintah DISTINCT di awal query SELECT, maka hanya data yang unik saja (data yang tidak sama) yang akan ditampilkan.
- ✓ Seandainya hasil query terdapat data yang sama lebih dari 1 kali tampil, perintah DISTINCT hanya akan menampilkannya 1 kali saja

```
SELECT DISTINCT nama_dosen,alamat
FROM daftar_dosen
ORDER BY alamat
```

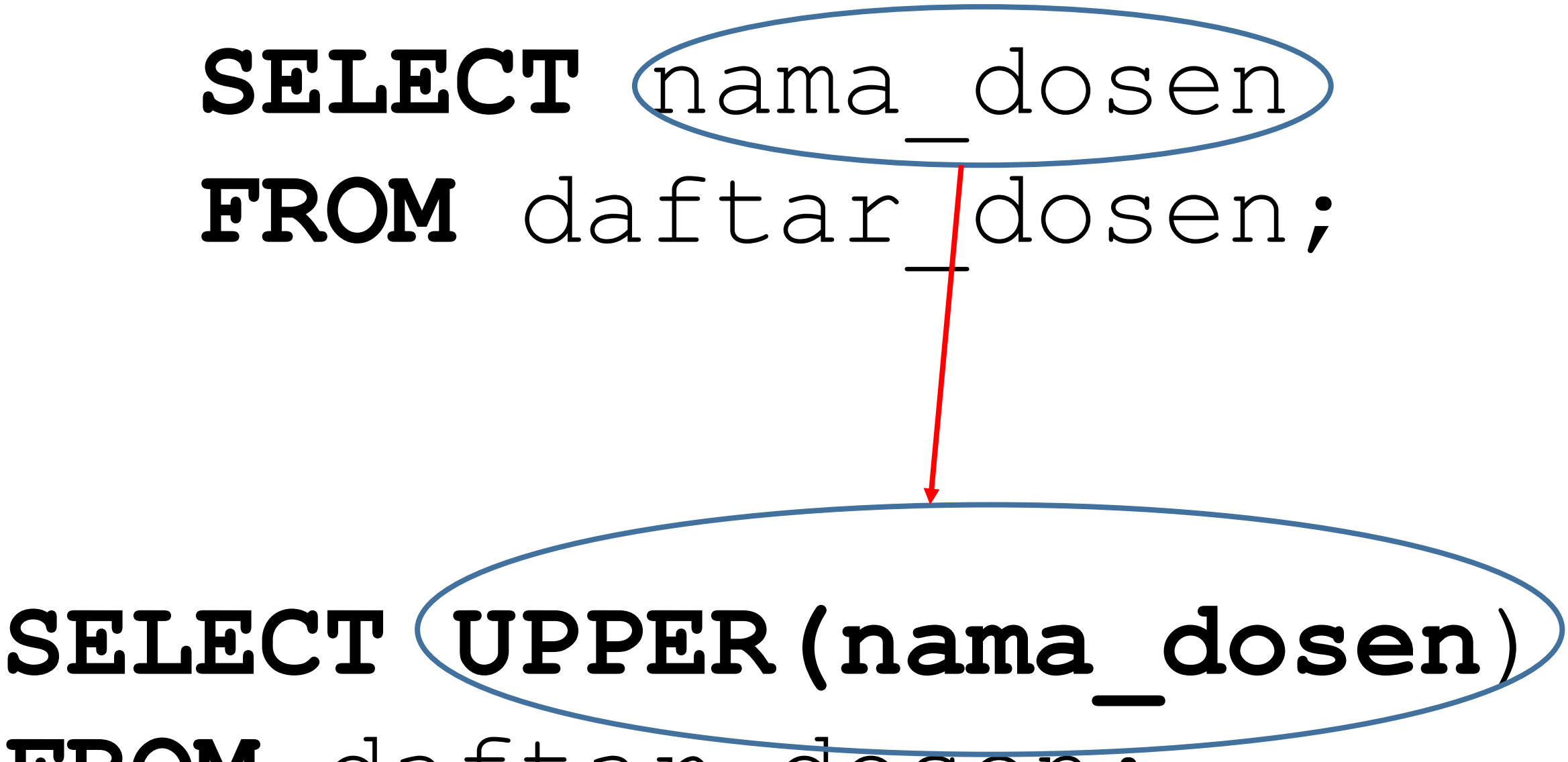
- ✓ MySQL tetap menampilkan seluruh isi tabel tanpa ada yang dieliminasi. Hal ini dikarenakan query DISTINCT hanya mengeleminasi query yang unik, atau tidak sama dilihat secara baris per baris (per record).

- ✓ Dengan mengkombinasikan nama_dosen dengan alamat, maka setiap baris dianggap unik, kecuali terdapat nama dosen dan alamat yang persis sama.

UPPER CASE



```
SELECT UPPER(nama_field_yang_akan_di_upper)
FROM tabel;
```

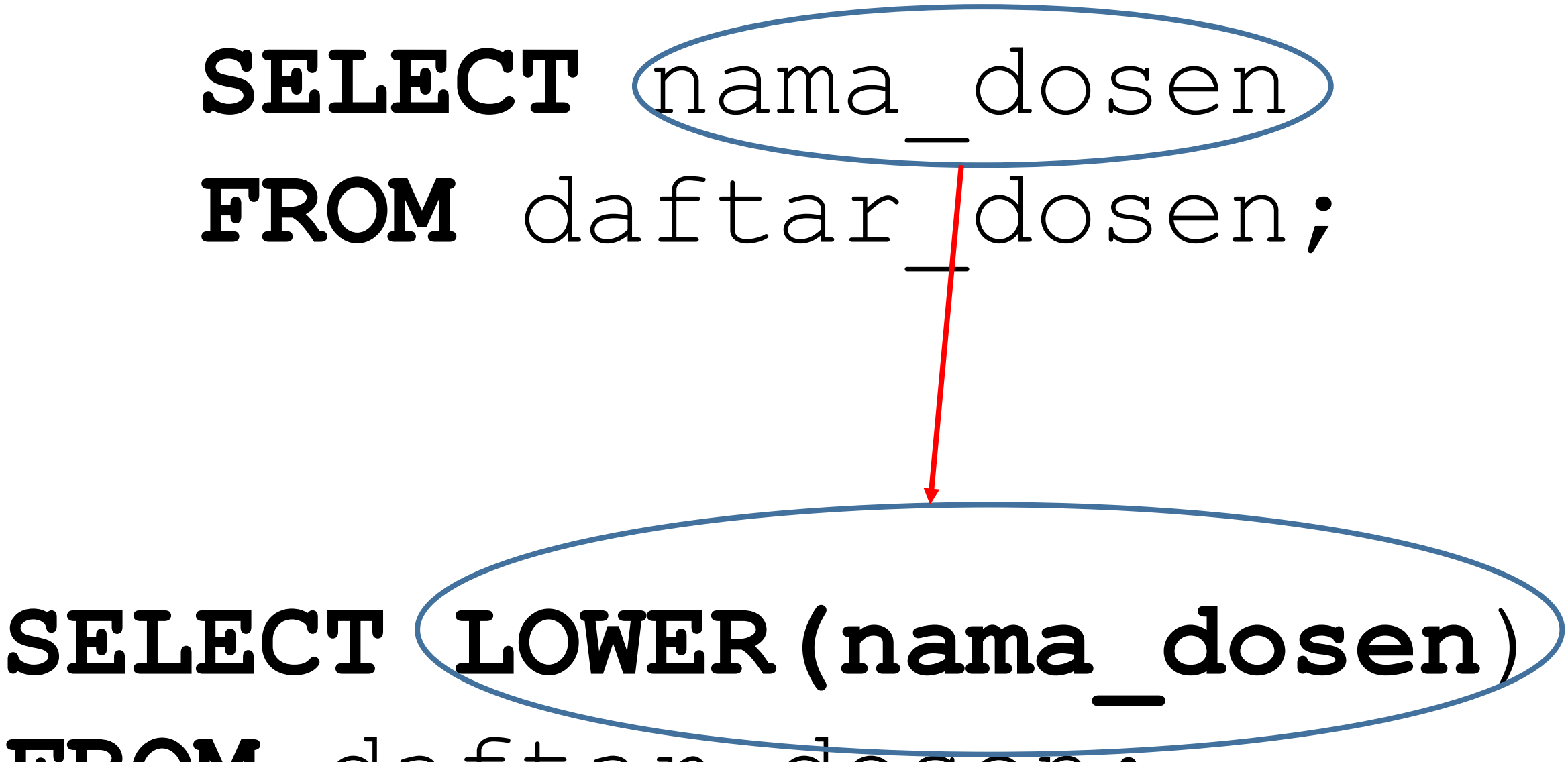

SELECT nama_dosen
FROM daftar_dosen;
SELECT UPPER(nama_dosen)
FROM daftar_dosen;

```
SELECT UPPER(nama_dosen), UPPER(alamat)
FROM daftar_dosen;
```

Jika ingin menampilkan kolom lain dari tabel yang sama? cukup dengan menyisipkan fungsi UPPER() ke dalam kolom yang ingin diubah

lower case

```
SELECT LOWER(nama_field_yang_akan_di_lower)
FROM tabel;
```


SELECT nama_dosen
FROM daftar_dosen;
SELECT LOWER(nama_dosen)
FROM daftar_dosen;

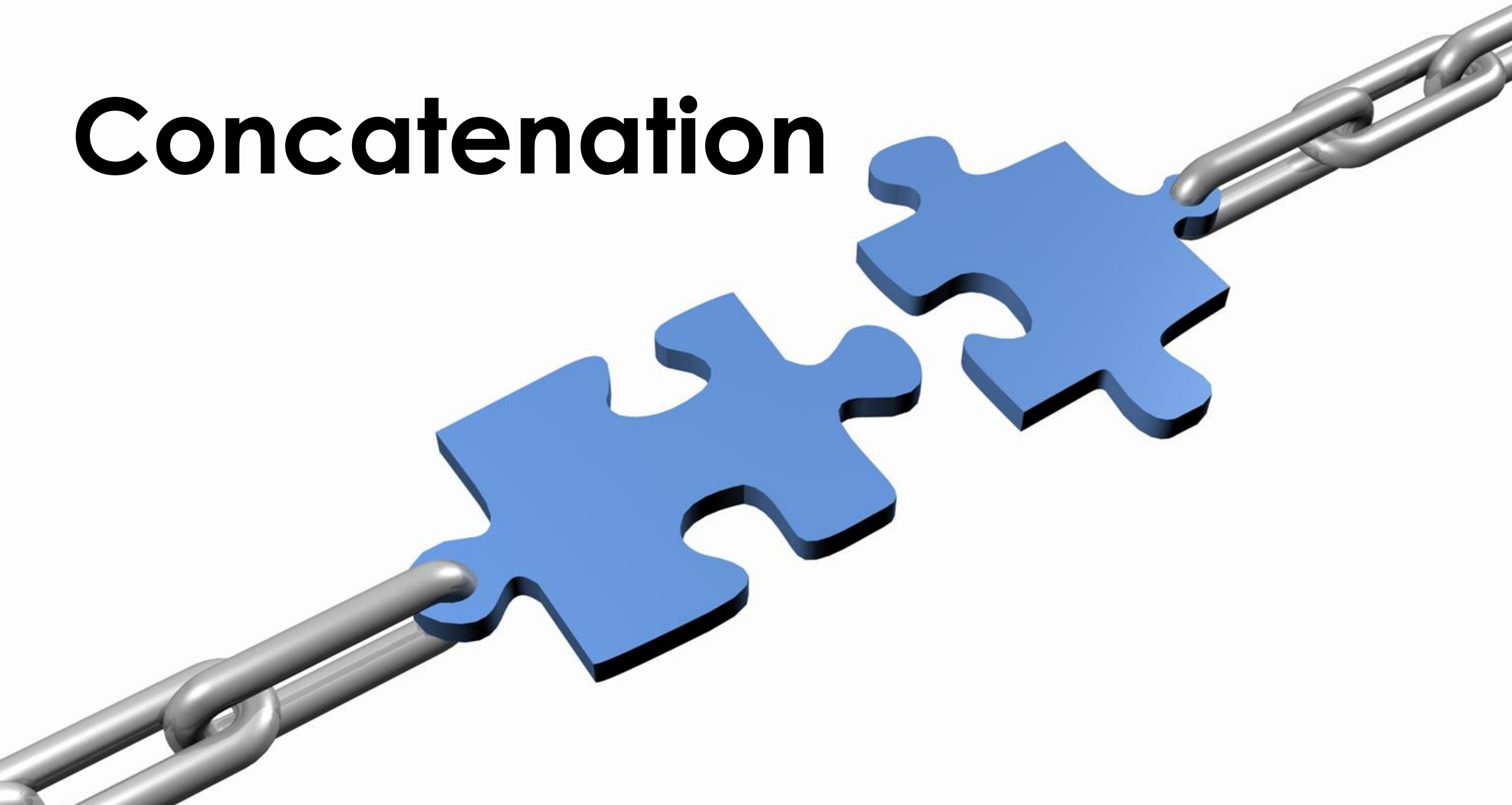
```
SELECT LOWER(nama_dosen), LOWER(alamat)
FROM daftar_dosen;
```

Jika ingin menampilkan kolom lain dari tabel yang sama? cukup dengan menyisipkan fungsi UPPER() ke dalam kolom yang ingin diubah

```
UPDATE daftar_dosen
SET nama_dosen = UPPER(nama_dosen);
```

Selain digunakan dalam menampilkan data (query SELECT), kita juga bisa menggunakan fungsi UPPER dan LOWER() untuk mengubah huruf di dalam kolom tabel secara permanen

Concatenation



```
SELECT CONCAT(field1, field2)
FROM nama_tabel;
```

MySQL memiliki fungsi bawaan yang bisa digunakan untuk menyambung string atau menggabungkan string hasil query, yakni melalui fungsi CONCAT() (singkatan dari concatenating)

```
SELECT CONCAT(nama_dosen,alamat)
FROM daftar_dosen;
```

```
SELECT CONCAT(nama_dosen,no_hp,alamat,nip_dosen)
FROM daftar_dosen;
```

Jika menggabungkan 3 atau 4 sekaligus? Kita tinggal menambahkan nama setiap kolom ke dalam fungsi CONCAT(), dan dipisahkan dengan karakter koma

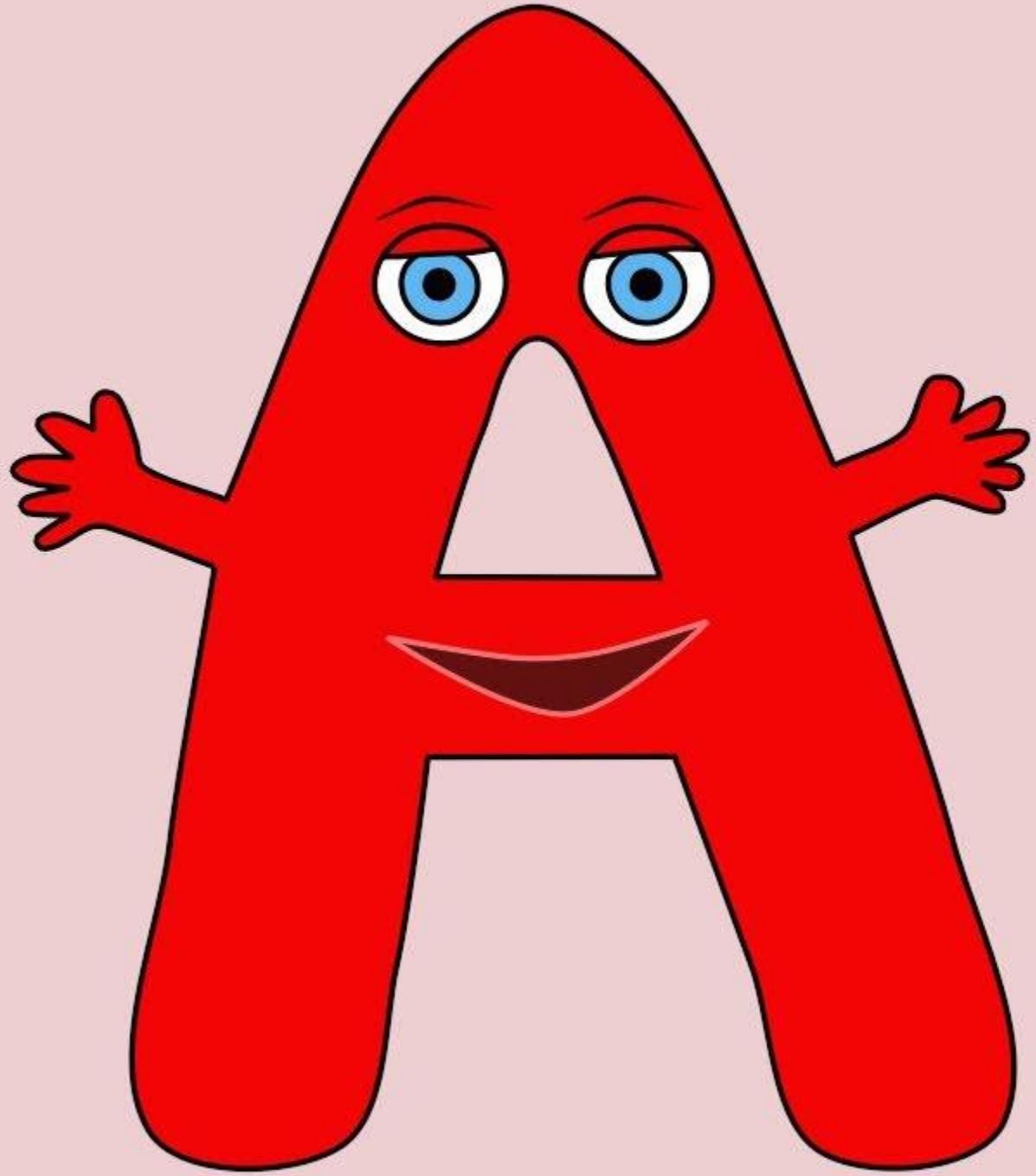
```
SELECT CONCAT_WS('karakter_tambahan', field1, field2, field-n)
FROM nama_tabel;
```

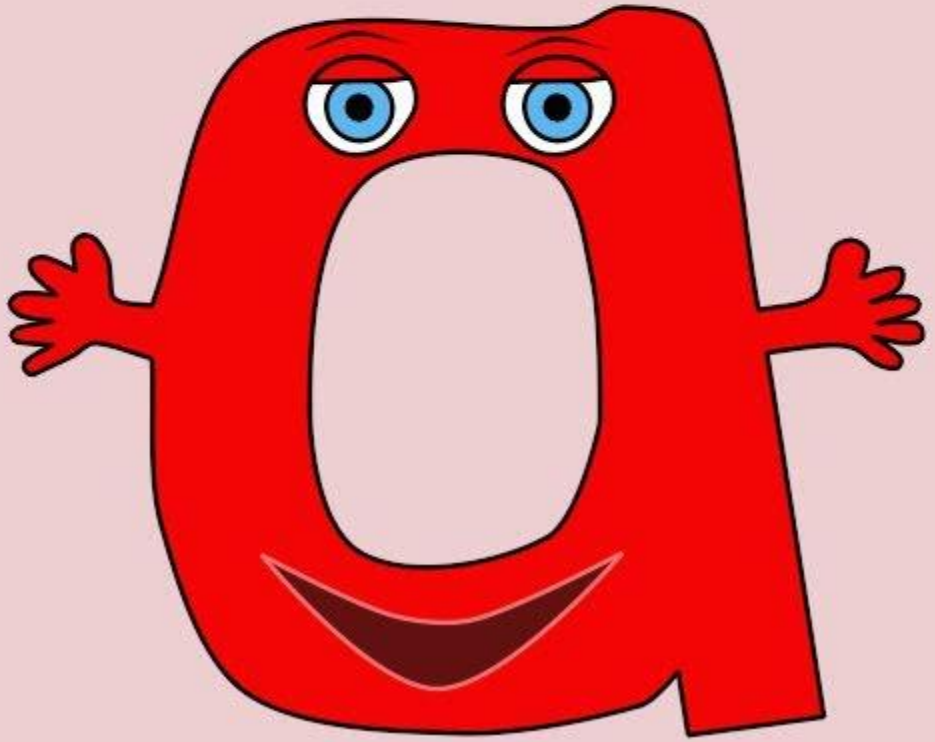
Fungsi CONCAT_WS() adalah variasi lain dai fungsi CONCAT(), perbedaannya dengan menggunakan fungsi CONCAT_WS(), dapat menambahkan karakter pembatas antara kolom yang akan digabung. Tambahan singkatan WS berarti ‘With Separator’

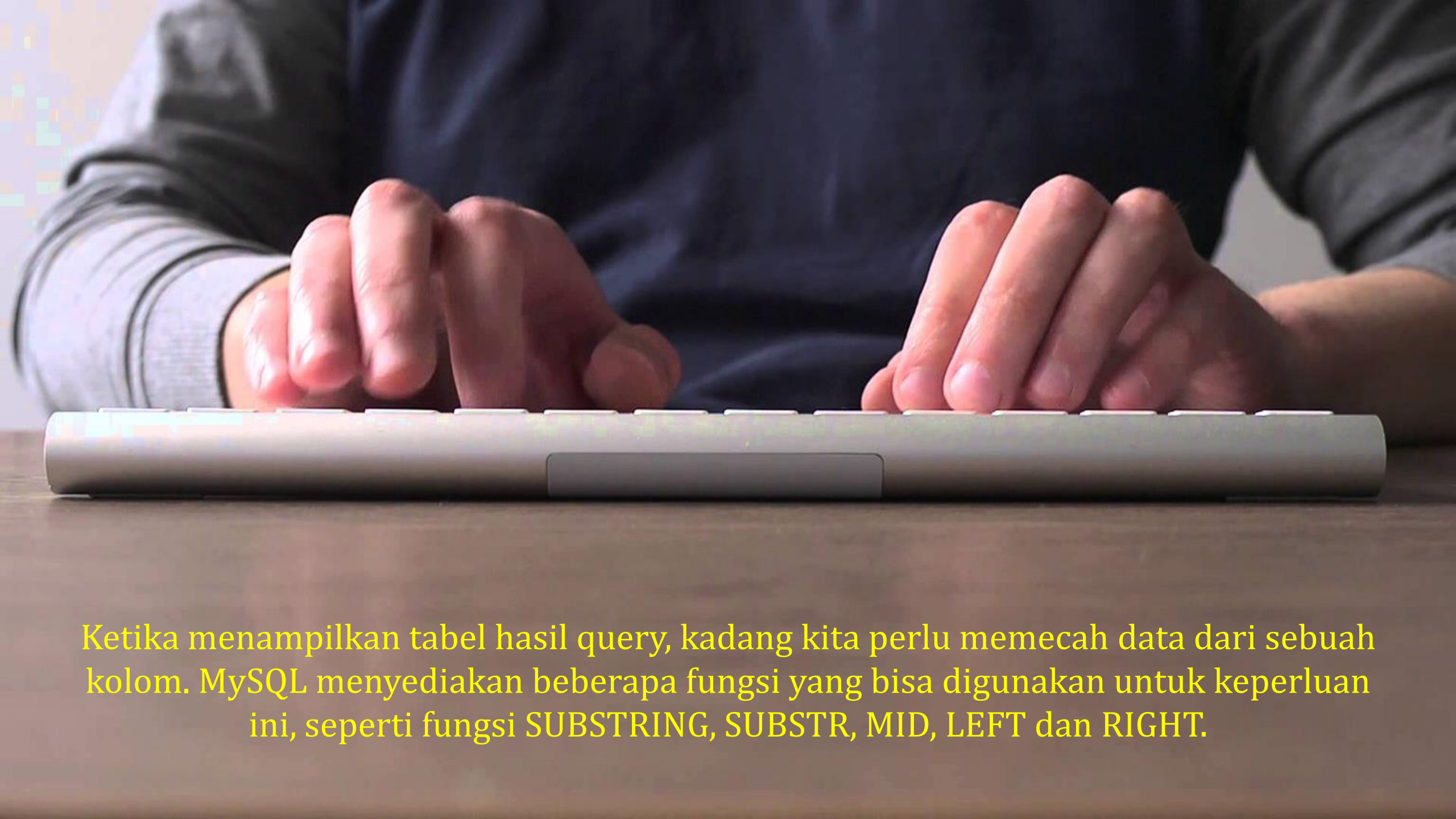
```
SELECT CONCAT_WS(' ',nama_dosen,alamat)
FROM daftar_dosen;

SELECT CONCAT(nama_dosen,' ',alamat)
FROM daftar_dosen;
```

```
SELECT CONCAT_WS(' ',nama_dosen,no_hp,alamat,nip_dosen)
FROM daftar_dosen;

SELECT CONCAT(nama_dosen,' ',no_hp,' ',alamat,' ',nip_dosen)
FROM daftar_dosen;
```

Substring







Ketika menampilkan tabel hasil query, kadang kita perlu memecah data dari sebuah kolom. MySQL menyediakan beberapa fungsi yang bisa digunakan untuk keperluan ini, seperti fungsi SUBSTRING, SUBSTR, MID, LEFT dan RIGHT.

SUBSTRING (nama_kolom, index_awal, jumlah_karakter)

nama_kolom

- nama kolom tabel yang akan diambil karakternya (sebagai sumber string)

index_awal

- bisa diisi dengan angka yang berfungsi sebagai index awal karakter yang ingin diambil, dihitung dari index 1 pada karakter pertama, index 2 pada karakter kedua, dst. Apabila diinput dengan nilai negatif, index akan dihitung mulai dari akhir string.

jumlah_karakter

- argumen opsional yang jika tidak ditulis, fungsi **SUBSTRING** akan mengambil seluruh karakter hingga akhir string. Jika ditulis, maka ini berfungsi sebagai batasan jumlah karakter yang akan diambil

```
SELECT NIP FROM daftar_dosen;
```

Tampilkan digit NIP terhitung dari digit ke-4

```
SELECT SUBSTRING(NIP, 4) FROM daftar_dosen;
```

Tampilkan 3 digit NIP terhitung dari digit ke-4 dari kiri

```
SELECT SUBSTRING(NIP, 4, 3) FROM daftar_dosen;
```

Tampilkan 4 digit terakhir NIP

```
SELECT SUBSTRING(NIP, -4) FROM daftar_dosen;
```

Tampilkan 3 digit NIP terhitung dari digit ke-4 dari kanan

```
SELECT SUBSTRING(NIP, -4, 3) FROM daftar_dosen
```

Dalam MySQL, fungsi SUBSTRING memiliki 2 alias: SUBSTR dan MID. Dengan kata lain, ketiga fungsi ini berfungsi sama.

```
SELECT SUBSTRING(NIP, -4, 3) FROM daftar_dosen;

SELECT SUBSTR(NIP, -4, 3) FROM daftar_dosen;

SELECT MID(NIP, -4, 3) FROM daftar_dosen;
```


LEFT
RIGHT

- ✓Fungsi LEFT dan RIGHT pada dasarnya merupakan bentuk sederhana dari fungsi SUBSTRING. Keduanya membutuhkan 2 argumen yaitu nama kolom tabel dan jumlah huruf yang ingin diambil.

- ✓Fungsi **LEFT** akan mengambil sejumlah karakter mulai dari **kiri** string, dan fungsi **RIGHT** akan mulai dari sisi **kanan** string

```
SELECT LEFT(NIP, 5) FROM daftar_dosen;

SELECT RIGHT(NIP, 5) FROM daftar_dosen;
```


Calendar
SUNDAY
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY

Tipe data tanggal (DATE) merupakan salah satu tipe data yang cukup sulit digunakan. Selain memerlukan format tersendiri, hasil dari kolom dengan tipe data date juga ‘terbalik’, yakni dengan format tampilan YYYY-MM-DD

```
SELECT YEAR(dt) FROM belajar_date;

SELECT MONTH(dt) FROM belajar_date;

SELECT DAY(dt) FROM belajar_date;
```

Nama Fungsi	Hasil
YEAR()	Menampilkan nilai tahun
MONTH()	Menampilkan nilai bulan (1..12)
MONTHNAME()	Menampilkan nama bulan (January..December)
DAYOFMONTH()	Menampilkan nilai tanggal (1..31)
DAYNAME()	Menampilkan nama hari (Sunday..Saturday)
DAYOFWEEK()	Menampilkan nilai hari (angka 1..7)
WEEKDAY()	Menampilkan nilai hari (angka 0..6)
DAYOFYEAR()	Menampilkan urutan hari (1..366)
HOUR()	Menampilkan nilai jam (0..23)
MINUTE()	Menampilkan nilai menit (0..59)
SECOND()	Menampilkan nilai detik(0..59)

```
SELECT HOUR(ts) FROM belajar_date;

SELECT MINUTE(ts) FROM belajar_date;

SELECT SECOND(ts) FROM belajar_date;

SELECT DAYOFYEAR(ts) FROM belajar_date;

SELECT DAYNAME(ts) FROM belajar_date;

SELECT ts, DAYNAME(ts) FROM belajar_date;
```

MySQL menyediakan fungsi bawaan untuk menghasilkan nilai DATE dan TIMESTAMP sistem saat ini. Fungsi yang bisa digunakan adalah CURDATE() dan NOW().

```
INSERT INTO belajar_date VALUES (CURDATE(), NOW());
```

Fungsi CURDATE() akan menghasilkan nilai DATE yang ada di server dengan format YYYY-MM-DD, sedangkan fungsi NOW() akan menghasilkan nilai TIMESTAMP dengan format YYYY-MM-DD HH:MM:SS.

```
SELECT NOW();
```

```
SELECT CURDATE();
```

Secara default, MySQL menyimpan tipe data tanggal (DATE) dengan format YYYY-MM-DD dan tipe data TIMESTAMP dengan format YYYY-MM-DD HH:MM:SS

```
DATE_FORMAT(nama_kolom, ‘string format’)

SELECT DATE_FORMAT(dt, '%d/%m/%Y‘)
FROM belajar_date;
```

Format	Penjelasan
%a	Abbreviated weekday name (Sun..Sat)
%b	Abbreviated month name (Jan..Dec)
%c	Month, numeric (0..12)
%D	Day of the month with English suffix (0th, 1st, 2nd, 3rd, ...)
%d	Day of the month, numeric (00..31)
%e	Day of the month, numeric (0..31)
%f	Microseconds (000000..999999)
%H	Hour (00..23)
%h	Hour (01..12)
%I	Hour (01..12)
%i	Minutes, numeric (00..59)
%j	Day of year (001..366)
%k	Hour (0..23)
%l	Hour (1..12)
%M	Month name (January..December)
%m	Month, numeric (00..12)
%p	AM or PM
%r	Time, 12-hour (hh:mm:ss followed by AM or PM)
%S	Seconds (00..59)
%s	Seconds (00..59)
%T	Time, 24-hour (hh:mm:ss)
%U	Week (00..53), where Sunday is the first day of the week; WEEK() mode 0
%u	Week (00..53), where Monday is the first day of the week; WEEK() mode 1
%V	Week (01..53), where Sunday is the first day of the week; WEEK() mode 2; used with %X
%v	Week (01..53), where Monday is the first day of the week; WEEK() mode 3; used with %x
%W	Weekday name (Sunday..Saturday)
%w	Day of the week (0=Sunday..6=Saturday)
%X	Year for the week where Sunday is the first day of the week, numeric, four digits; used with %V
%x	Year for the week, where Monday is the first day of the week, numeric, four digits; used with %v
%Y	Year, numeric, four digits
%y	Year, numeric (two digits)
%%	A literal “%” character
%x	x, for any “x” not listed above

10 - 08 - 2015
17 - 08 - 2016
31 - 12 - 2017

```
SELECT DATE_FORMAT(dt, '%d - %m - %Y')
FROM belajar_date;
```

10 August 2015
17 August 2016
31 December 2017

```
SELECT DATE_FORMAT(dt, '%d %M %Y')
FROM belajar_date;
```

```
SELECT DATE_FORMAT(ts, '%d %M %Y, %k:%i:%s')
FROM belajar_date;
```

10 August 2015, 8:30:15
17 August 2016, 10:01:01
31 December 2017, 23:59:59