



Cari Slide

Tim



ooowwwIII(Lv.26)



- Node.js
- React
- SQL 4

- Semua
- I
- II
- III
- IV

- Python

- Semua
- I
- II
- III
- IV
- V

SQL

Semua

Favorit

### Ayo Belajar SQL!

Dalam pelajaran ini, Anda akan mempelajari SQL sebagai bahasa untuk mengolah database. Setelah mempelajari ini, Anda akan bisa memanfaatkan segera bidang karir dan database bersifat universal.



### Database

Database adalah tempat untuk menyimpan string dan angka. Contohnya, seluruh data 'postingan' media sosial seperti Twitter dan 'belajar' di Progate akan disimpan dalam database. ('string' adalah sedikit karakter)



### Tabel untuk Pelajaran ini

Untuk pelajaran ini, kita akan menggunakan tabel bernama purchases yang berisi data tentang pesanan sejuta yang dibuat oleh karakter-karakter Progate. Mari kita lihat lebih detail melalui latihan di bawah berikutnya.

#### Tabel purchases

Nama Kolom	Data Disimpan
id	nomor ID
name	nama barang
price	harga barang
character_name	nama karakter
category	kategori barang
purchased_at	tanggal

### SELECT

Dalam SQL, kita menggunakan klausus SELECT untuk mendapatkan data dari database. SELECT digunakan untuk memilih dan mengambil kolom data yang ingin anda lihat.

### Menyelesaikan Kueri SQL

Untuk mempelajari statement SQL, bukan sekedar tahu di depan statement tersebut. Anda dapat memulai statement SQL di sini atau baris, lalu apapun di mana pun, karena baris baru untuk ketika perintah tidak case-sensitive. Anda bisa untuk menggunakan huruf besar atau kecil tetapi tidak sakitan apabila mengikuti format penulisan seperti gambar di bawah.

### Mengambil Data dari Beberapa Kolom

Jika Anda ingin mengambil data dari beberapa kolom, gunakan koma untuk memisahkan setiap nama kolom, seperti ditunjukkan ini:

### Analisis Data dengan SQL



### Struktur Database

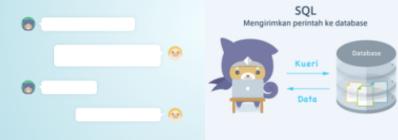
Database mengelola data didalam tabel seperti gambar di bawah. Secara vertical, data disusun dalam kolom dan secara horizontal, data disusun dalam baris. Didalam database, kita bisa membuat banyak tabel.

Kolom	Baris	data
id	1	puuting
name	2	pena
price	3	buku kali
character_name	4	puuting
category		1
purchased_at		2018-10-10



Hubungi kami!

### Apa itu Kueri?



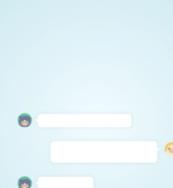
### FROM

Biasanya setiap database terdiri dari beberapa tabel. Karena inilah, kita harus menggunakan klausus FROM untuk memilih spesifik tabel tersebut. Dengan menggunakan SELECT & FROM, maka kita dapat secara spesifik memilih kolom tertentu dari tabel tertentu untuk dianalisa dan diolah.



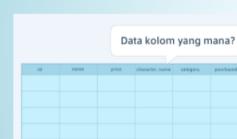
Hubungi kami!

### Memilih Banyak Kolom



### Memilih dari Semua Kolom

Jika Anda ingin mendapatkan data dari semua kolom dalam tabel, gunakan simbol '\*'.



## Mendapatkan Baris Tertentu



Mari kita dapatkan data semua baris "makanan"					
ID	Name	Price	Character Name	Category	Purchased At
1	pudding	2	Ninjia Ken	makanan	2018-10-10
2	biskuit	1	lalimya	makanan	2018-10-10
3	buah buah	2	lalimya	makanan	2018-10-10
4	pudding	1	Ninjia Ken	makanan	2018-10-10

## WHERE

Untuk mendapatkan baris tertentu, gunakan klausus WHERE. Sampai sekarang, kita sudah mempelajari cara untuk mengakses kolom di tabel tertentu untuk mendapatkan baris-barisnya. Dengan menggunakan WHERE ini, dapat secara spesifik menunjukkan baris yang memiliki data tertentu.

[Hubungi kami!](#)

## WHERE (lanjutan)

Gunakan tanda = seperti contoh di bawah ini untuk menunjukkan baris mana yang ingin Anda ambil dari tabel dalam database.

```
sample.sql >
SELECT *
FROM purchases
WHERE category = "makanan"
dapatkan baris dimana kolom category bernilai "makanan"
```

ID	Name	Price	Category
1	pudding	2	makanan
2	pasta	1	lalimya
3	buah buah	2	lalimya
4	pudding	1	makanan

## Mengambil Baris "Makanan"

Dalam klausus WHERE kita akan berusaha mencari baris yang nilai kolom "category" ini adalah "makanan". Jangan lupa akan segera tampil hasil dari query SQL. Data tersebut bisa seperti "makanan" harus dimasukkan dalam tanda kutip. Kita akan belajar tentang teknik data yang ada dalam pelajaran ke depannya.

```
sample.sql >
SELECT *
FROM purchases
WHERE category = "makanan"
gunakan tanda kutip
```

ID	Name	Price	Character Name	Category	Purchased At
1	pudding	2	Ninjia Ken	makanan	2018-10-10
2	biskuit	1	lalimya	pernak-pernik	2018-10-10
3	salat	4	Guru Besar	makanan	2018-10-10
4	roti gading	5	Akula	makanan	2018-10-10
5	buah buah	4	Guru Besar	pernak-pernik	2018-10-10

## Beberapa Kolom dengan Kondisi



## Jenis Data (lanjutan)

Setiap kolom dalam tabel memiliki jenis data yang berbeda. Jenis data tabel yang akan digunakan dalam pelajaran ini dapat dilihat dari gambar di bawah ini.

Kolom	Tipe Data	Data
id	angka	1
name	teks	"rantangan"
price	angka	10
character_name	teks	"Ninjia Ken"
category	teks	"makanan"
purchased_at	tanggal	"2018-10-10"

## Data Tanggal

Selanjutnya, kita akan menampilkan baris dengan tanggal "2018-10-10". Data dalam kolom "purchased\_at" disimpan sebagai jenis data "tanggal". Untuk jenis data tanggal, diperlukan tanda kutip. Jadi, jangan lupa menulis tanda kutip pada setiap penulisan nilai data tanggal!

```
sample.sql >
SELECT *
FROM purchases
WHERE purchased_at = "2018-10-10";
gunakan tanda kutip untuk tanggal
```

## Menampilkan Data Dengan Harga "\$10 atau lebih"



ID	Name	Price	Category
1	pudding	2	makanan
2	cacing smartphone	10	pernak-pernik
3	buah	30	lalimya
4	pudding	1	makanan

## Jenis Data

Ada peraturan menampilkan data di dalam database, namanya jenis data. Masing-masing untuk menyatakan berbagai jenis data seperti teks, angka (integer/number), dan bahkan tanggal (date).

[Hubungi kami!](#)

```
sample.sql >
SELECT *
FROM purchases
WHERE price = 10;
tanpa tanda kutip!
```

## Data Angka

Mari kita coba menulis buku untuk menampilkan semua baris yang memenuhi kriteria "produk dengan harga \$10". Harga dalam kolom "price" disimpan sebagai jenis data integer. Ingatlah, untuk jenis data angka/integer Anda tidak perlu menggunakan tanda kutip.

[Hubungi kami!](#)

## Tinjauan Ulang Jenis Data

Kita harus mempertimbangkan beberapa karakteristik setiap jenis data. Tadi, untuk pelajaran ini, Anda hanya perlu mengingat 3 jenis data seperti yang ditampilkan dalam gambar di bawah.

Catatan tentang tipe data		
Tipe Data	Contoh	Karakteristik
teks	"Ninjia Ken"	teks diberikan tanda kutip
angka	10	angka (tanpa tanda kutip)
tanggal	"2018-10-10"	tanggal diberikan tanda kutip (format: tahun-bulan-hari)

## Operator Perbandingan

Selain tanda =, terdapat simbol lain yang dapat digunakan dalam klausus WHERE. Merelya disebut operator perbandingan. Seperi operator lebih dari dan kurang dari pada tanda garis dibuat. Dengan ini, Anda dapat membuat kondisi seperti "harga buku lebih dari atau sama dengan \$10" untuk menampilkan baris dengan cara yang mudah.

Contoh Operator Perbandingan	
a < b	... mencari a lebih kecil dari b
a <= b	... mencari a lebih kecil dari atau sama dengan b
a > b	... mencari a lebih besar dari b
a >= b	... mencari a lebih besar dari atau sama dengan b

```
sample.sql >
SELECT *
FROM purchases
WHERE price >= 10;
caril baris dengan harga yang "lebih besar dari atau sama dengan" 10
```

## Operator Perbandingan (lanjutan)

Selain data angka dalam kolom "price", operator perbandingan dapat digunakan pada jenis data "tanggal" seperti pada kolom "purchased\_at". Jangan lupa gunakan tanda kutip untuk nilai tanggalnya!

```
sample.sql >
SELECT *
FROM purchases
WHERE purchased_at <= "2018-11-01";
caril baris dengan tanggal "sebelum atau sama dengan 2018-11-01"
```

## Baris yang Berisi "Puding"

pencarian sebelumnya	
WHERE name = "pudding"	
--> dapat mengambil date int	
--> tidak dapat mengambil date int	

hanya mengambil baris yang berisi-beri sama dengan kondisi yang ditentukan

[Hubungi kami!](#)

## Operator LIKE

Saat ingin memperiksa baris yang berisi karakter tertentu, gunakan operator LIKE. Sebagaimana ditampilkan di bawah, kamu bisa menetapkan kondisi untuk menampilkan 'baris yang berisi string tertentu' di kolom yang dituju.

```
sample.sql
SELECT *
FROM purchases
WHERE name LIKE nilai_string;
```

## Menggunakan LIKE dengan Wildcard



## Pencarian Postfiks

Kamu juga bisa mencari nilai dengan karakter string tertentu, misalnya '%puing'. (hat contoh dibawah ini). Sama halnya yang diajarkan dengan string 'puing' akan diimplementasikan. Pencarian sebenarnya ini dibuat pengetahuan postfix.

```
Pencarian postfiks
sample.sql
SELECT *
FROM purchases
WHERE name LIKE "%puing";
mencari nilai yang berakhir dengan "puing"
```

- puding.
- ✗ puding susu
- kue puding.
- ✗ puding panggang mochi

## Operator NOT

Untuk menampilkan baris yang tidak memiliki nilai tertentu, gunakan operator NOT. Selainnya, NOT bisa digunakan dalam semua operasi yang sudah kita pelajari. Cengan begitu, semua baris yang tidak memenuhi kondisi yang ditentukan akan ditampilkan.

```
sample.sql
SELECT *
FROM purchases
WHERE NOT price > 10;
mengambil baris yang tidak memenuhi kondisi
```

```
sample.sql
SELECT *
FROM purchases
WHERE NOT name LIKE "%puing%";
mengambil baris yang tidak memenuhi kondisi
```

## Apa itu NULL?



purchases			
id	name	price	urchased_at
1	puing	2	2018-10-10
2	pena	1	2018-10-10
3	buku tulis	2	2018-10-10
4	blanya buks	1	2018-10-10

## Menampilkan Baris Tanpa Nilai NULL

Sebaliknya, IS NOT NULL dapat digunakan untuk menampilkan baris yang tidak berisi nilai NULL untuk kolom tertentu.

```
sample.sql
SELECT * FROM purchases
WHERE price IS NOT NULL;
mengambil data dari kolom price yang isi barisnya bukanlah NULL
```

purchases			
id	name	price	urchased_at
1	puing	2	2018-10-10
2	pena	1	2018-10-10
3	buku tulis	2	2018-10-10
4	blanya buks	1	2018-10-10

## Mencari Makanan Guru Domba



## Wildcard

Untuk memperluas operator LIKE, Anda harus belajar tentang wildcard. Dalam dunia pemrograman, '%wildcard' adalah simbol untuk menggantikan satu (ataupun) karakter dalam sebuah string. Simbol wildcard (%) dapat digunakan dengan operator LIKE. Berikut ini adalah contoh menggunakan wildcard untuk menampilkan semua baris yang berisi 'buding' di kolom 'name'.

```
sample.sql
SELECT *
FROM purchases
WHERE name LIKE "%puing%";
```

Anda dapat mencari string apapun yang memiliki "puing" dengan memberikan wildcard (%) sebelum dan sesudah kata "puing".

- puding
- puding susu
- kue puding
- kue puding durian

## Pencarian Prefiks

Wildcard bisa digunakan di awal dan akhir sebuah string. Seperti di bawah ini, wildcard '%puing%' bisa digunakan untuk mencari "puing" dengan semua nilai di database yang dimulai dengan "puing" (Itu contoh dibawah). Pencarian ini disebut pencarian prefiks.

```
Pencarian prefiks
sample.sql
SELECT *
FROM purchases
WHERE name LIKE "puing%";
```

mencari nilai yang mulai dengan "puing".

- puding
- puding susu
- ✗ kue puding
- puding panggang mochi

[Hubungi kami!](#)

## Menggunakan Kondisi Negatif



Mengambil baris yang tidak memiliki kata "puing".			
id	name	price	urchased_at
1	puing	2	2018-10-10
2	pena	1	2018-10-10
3	buku tulis	2	2018-10-10
4	puing	1	2018-10-10

## Baris dengan Kolom NULL



## Menampilkan Baris dengan Nilai NULL

Untuk memilih baris yang berisi nilai NULL di dalamnya, gunakan IS NULL. Sebagai contoh, dengan kondisi ini 'name IS NULL', kita bisa mencari baris dimana kolom 'name' berisi NULL. Ini berguna di situasi dimana kita mau mencari baris yang kita tahu pasti tanpa sengaja.

```
sample.sql
SELECT * FROM purchases
WHERE price IS NULL;
mengambil data dari kolom price yang isi barisnya adalah NULL
```

purchases			
id	name	price	urchased_at
1	puing	2	2018-10-10
2	pena	1	2018-10-10
3	buku tulis	2	2018-10-10
4	blanya buks	1	2018-10-10

Untuk mendapatkan baris dimana kolom tertentu, baris atau tidak berisi NULL, kamu tidak dapat menggunakan operator =. Hati-hati jangan sampai melupakan kesalahan ini.

```
sample.sql
SELECT * FROM purchases
WHERE price = NULL;
tidak bisa digunakan!
```

```
sample.sql
SELECT * FROM purchases
WHERE NOT price = NULL;
tidak bisa digunakan!
```

## Operator AND

Dengan menggunakan AND, kita bisa menentukan beberapa kondisi atau syarat untuk klausus WHERE. Gunakan struktur 'WHERE kondisi1 AND kondisi2' untuk mencari baris yang memenuhi kedua kondisi.

```
sample.sql
SELECT * FROM purchases
WHERE character_name = "Guru Domba"
AND category = "makanan";
```

purchases			
id	name	category	character_name
1	buku	IaIanya	Guru Domba
2	puing	makanan	Briele
3	buku tulis	IaIanya	Ninja Ken

[Hubungi kami!](#)

## Operator OR

Seperi operator AND, operator OR dapat digunakan di kueri SQL untuk memasukkan beberapa kondisi. Gunakan struktur "WHERE kondisi1 OR kondisi2" untuk mendapatkan baris yang memenuhi salah satu kondisi (atau kedua).

```
sample.sql >
SELECT * FROM purchases
WHERE character_name = 'Guru Domba'
    kondisi1
OR character_name = 'Ninja Ken'
    kondisi2
```

purchases			
ID	Name	Category	Character Name
1	buku	lainnya	Guru Domba
2	puding	makanan	Birde
3	buku tulis	lainnya	Ninja Ken
4	puding	makanan	Guru Domba

## Mengurutkan Baris

ID	Name	Price	Character Name
7	Casing smartphone	15	Ninja Ken
2	antingan	5	Guru Domba
5	buku tulis	2	Ninja Ken
1	teh	2	Birde
6	pena	1	Ninja Ken
4	puding	1	Birde
3	puding	1	Ninja Ken

mengurutkan baris berdasarkan price secara menurun.

## ORDER BY (1)

Dalam SQL, kamu bisa mengurutkan baris menggunakan ORDER BY. Kamu juga bisa memilih kolom tertentu yang ingin kamu urutkan, seperti contoh berikut ini. Kamu juga harus menentukan bagaimana cara pengurutannya di bagian akhir kueri.

```
sample.sql >
ORDER BY column_name method_order;
```

## ORDER BY (2)

Jika menuliskan ORDER BY di akhir sebuah statement SQL, hasil akan diurutkan seperti contoh di bawah ini.

```
sample.sql >
SELECT *
FROM purchases
ORDER BY price DESC;
```

ORDER BY price DESC;			
ID	Name	Price	published_at
7	casing smartphone	15	2018-10-10
2	antingan	5	2018-10-10
5	buku tulis	2	2018-10-10
1	teh	2	2018-10-11
6	pena	1	2018-10-11
4	puding	1	2018-10-11
3	puding	1	2018-10-12

mengurutkan baris berdasarkan price secara menurun.

## Urutan Naik & Turun

ORDER BY memerlukan method untuk mengurutkan data secara "menurun (DESC)" atau "menaik (ASC)". Untuk naikkan nilai terkecil terlebih dahulu. Sedangkan urutan turun akan metekan nilai terbesar terlebih dahulu. Dalam code SQL, ASC digunakan untuk "ascending (menaik)" sedangkan DESC digunakan untuk "descending (menurun)".

ASC(menaik) : 1,2,3...100  
terkecil → terbesar

DESC(menurun) : 100...3,2,1  
terbesar → terkecil

✉ Hubungi kami!

## Hanya Menampilkan Baris yang Diperlukan



## ORDER BY (3)

Karena ORDER BY digunakan di dalam statement SQL, ORDER BY dapat digunakan dengan klausus WHERE seperti gambar dibawah.

```
sample.sql >
SELECT *
FROM purchases
WHERE kondisi
ORDER BY price DESC;
```

Untuk implementasi "jumlah maksimum hasil" tertentu, gunakan LIMIT.

Untuk cara kerjanya pada contoh di bawah ini.

✉ Hubungi kami!

## LIMIT (1)

Untuk implementasi "jumlah maksimum hasil" tertentu, gunakan LIMIT.

Untuk cara kerjanya pada contoh di bawah ini.

ID	Name	Price	published_at
1	puding	1	2018-10-10
2	pena	1	2018-10-10
3	buku tulis	2	2018-10-10
4	puding	1	2018-10-11
5	teh	2	2018-10-11
6	antingan	5	2018-10-11
7	casing smartphone	15	2018-10-12

## LIMIT (2)

Seperi ORDER BY LIMIT digunakan di akhir sebuah kueri SQL.

```
sample.sql >
SELECT *
FROM purchases
LIMIT 5;
```



## LIMIT (3)

Selain itu, seperti ORDER BY, LIMIT juga bisa digunakan bersama dengan klausus WHERE.

```
sample.sql >
SELECT *
FROM purchases
WHERE kondisi
LIMIT 5;
```

ORDER BY dan LIMIT dibutuhkan kueri SQL, tapi mereka dapat digunakan bersamaan dalam satu pernyataan. Pada saat menggunakan keduanya secara bersamaan, LIMIT harus dibuat seiring setelah ORDER BY.

## Tinjauan Ulang Pelajaran



**[Tambah]**  
Menggabungkan ORDER BY dengan LIMIT

Untuk cara kerjanya pada contoh di bawah ini.

```
sample.sql >
SELECT *
FROM purchases
ORDER BY price DESC
LIMIT 5;
```

✉ Hubungi kami!



## Ayo Belajar SQL Secara Praktis

Dalam pelajaran ini, kita akan mempelajari teknik SQL yang lebih praktis dan berguna. Untuk mempermudah Anda dalam mengolah dan menganalisa data yang lebih rumit.

Total	Tanggal
3	2018-10-10

**Mengecualikan Data Duplikat**

Sebagaimana ditemukan di bawah, Anda dapat menggunakan DISTINCT dalam perintah SELECT untuk mengecualikan baris duplikat. Dalam contoh dibawah, perintah memiliki kolom name dari tabel purchases dengan mengecualikan duplikat.

```
sample.sql
SELECT DISTINCT(name)
FROM purchases;
```

**DISTINCT**

Dengan menggunakan DISTINCT, Anda dapat mengecualikan baris yang memiliki data yang sama dari hasil pencarian. Anda dapat menentukan nama kolom untuk mendapatkan baris unik dan mengeliminasi duplikatnya. Untuk melakukan hal ini, gunakan syntax berikut: DISTINCT(column\_name).

[Hubungi kami!](#)

**Menggunakan Operator Aritmetika**

Penghitungan dapat dilakukan di SQL dengan operator aritmetika. Sebagaimana ditemukan di bawah, dengan menggunakan operator ini, Anda dapat melakukan penghitungan di baris pada kolom yang Anda tentukan.

+	... penjumlahan
-	... pengurangan
*	... perkalian
/	... pembagian

```
sample.sql
SELECT name, price * 1.09
        gunakan operator aritmetika
FROM purchases;
```

**Operator Aritmetika**

Agar bisa mendapatkan harga termasuk pajak, kita akan mengalihkan kolom price dengan 1.09 (pajak sebesar 10%). Dalam kueri SQL, untuk mendapatkan nilai hasil penghitungan, operator aritmetika dapat digunakan setelah kata kunci SELECT.

[Hubungi kami!](#)

**Menggunakan Function**

Untuk menghitung jumlah angka di SQL, gunakan function SUM. Dengan syntax SUM(column\_name), dengan perintah ini, penghitungan total semua nilai dalam kolom yang ditentukan bisa dilakukan.

```
sample.sql
SELECT SUM(column_name)
```

**Mengumpulkan Jumlah Total**

Function SUM digunakan setelah SELECT untuk mendapatkan hasil agregat. Dengan kueri seperti ini, sebuah contoh hasilnya:

```
sample.sql
SELECT SUM(price)
FROM purchases;
```

[Hubungi kami!](#)

**SUM**

Penggunaan function SUM juga dapat dikombinasikan dengan WHERE. Pada contoh di bawah, headmania dapat dihitung total untuk mendapatkan jumlah total yang digunakan oleh Ninja Ken.

```
sample.sql
SELECT SUM(price) ② dapatkan jumlah total dari
                seluruh price dari headmania
                pesanan yang dimiliki oleh
                character_name = "Ninja Ken";
③ dapatkan baris dengan nilai "Ninja Ken"
```

**Menghitung Rata-Rata**

Function AVG dapat digunakan setelah SELECT untuk menghitung rata-rata baris untuk kolom tertentu. Di kueri bawah ini adalah contoh kueriannya.

[Hubungi kami!](#)

**AVG**

Untuk menghitung rata-rata angka di SQL, Anda dapat menggunakan AVG. Menggunakan syntax AVG(column\_name), Anda dapat menghitung nilai rata-rata pada kolom yang ditentukan.

```
sample.sql
```

**Menggunakan AVG**

Function AVG dapat digunakan setelah SELECT untuk menghitung rata-rata baris untuk kolom tertentu. Di kueri bawah ini adalah contoh kueriannya.

[Hubungi kami!](#)

**AVG(column\_name)**

ID	Name	Price
1	puding	1
2	pena	1
3	teh	2
4	puding	2
5	buku tulis	2

Dapatkan nilai rata-rata dari kolom yang telah disepesifikasi

**SELECT AVG(price) FROM purchases;**

Avg(price)
4

Menggabungkan WHERE & AVG

Penggunaan fungsi AVG juga dapat dikombinasikan dengan klausus WHERE. Pada contoh di bawah, kita menggunakan WHERE untuk mendasarkan harga rata-rata pembelian yang dimusnahi dilakukan oleh Ninja Ken.

```
sample.sql
SELECT AVG(price) ② Kalkulasikan rata-rata kolom price dari hasil peringatan baris dibawah
FROM purchases
WHERE character_name = "Ninja Ken";
① Cari baris dengan nilai "Ninja Ken"
```

Menghitung Jumlah Baris

ID	Name	Price
1	puding	1
2	pena	1
3	teh	2
4	puding	2
5	buku tulis	2

Hasil: Dapatkan jumlah total baris

**Hubungi kami!**

**COUNT**

Fungsi COUNT digunakan untuk menghitung jumlah total baris pada kolom yang ditargetkan. Ini dapat dilakukan dengan menggunakan sintaks COUNT(column\_name), sebagai ikhtiraam dalam contoh di bawah.

```
sample.sql
COUNT(column_name)
```

ID	Name	Price
1	puding	1
2	pena	1
3	teh	2
4	puding	2
5	buku tulis	2

Hasil: Dapatkan jumlah total baris didalam kolom!

**COUNT dan NULL**

Saat menggunakan COUNT, jumlah baris yang terhitung tidak mencakup baris dengan nilai NULL. Oleh karena itu, untuk kasus seperti gambar di sebelah kanan, hasil akhir hitungan COUNT adalah 4.

```
sample.sql
COUNT(price)
```

ID	Name	Price
1	puding	1
2	pena	1
3	teh	2
4	puding	2
5	buku tulis	2

Hasil: Baris dengan nilai NULL tidak dihitung

**Hubungi kami!**

Menggunakan COUNT

Jika Anda ingin menghitung semua baris, termasuk baris dengan nilai NULL, Anda harus menggunakan \* (tanda bintang) dengan fungsi COUNT. Ketika digunakan \*, akan menghitung jumlah total baris secara keseluruhan, termasuk yang nilainya NULL.

```
sample.sql
SELECT COUNT(*)
FROM purchases;
```

Hasil
COUNT( * )

Hasil: Menghitung jumlah total baris

**WHERE & COUNT**

Penggunaan fungsi COUNT juga dapat dikombinasikan dengan WHERE. Pada contoh di bawah, kita menggunakan keduanya untuk melihat total pembelian yang dilakukan oleh Ninja Ken.

```
sample.sql
SELECT COUNT(*)
FROM purchases
WHERE character_name = "Ninja Ken";
① Cari baris dengan nilai "Ninja Ken"
```

**Hubungi kami!**

Menemukan Nilai Maksimum dan Minimum

**MAX & MIN**

Dengan fungsi MAX di SQL, Anda bisa mendapatkan nilai maksimum dari baris milik kolom tertentu. Sebaliknya, dengan menggunakan function MIN, Anda bisa mendapatkan nilai minimum-muinya.

```
sample.sql
① MAX(column_name)
② MIN(column_name)
```

Hasil
MAX(price)

Hasil: 15

Hasil
MIN(price)

Hasil: 1

Menggunakan MAX dan MIN

Sama dengan function agregat lainnya, MAX dan MIN dapat digunakan setelah SELECT. Seperti contoh di bawah, dengan menekan kolom price di function MAX, Anda bisa mendapatkan item dengan harga tertinggi untuk semua baris kolom price.

```
sample.sql
SELECT MAX(price)
FROM purchases;
```

Hasil
-------

Hasil: MAX(price)

Hasil: 15

Menggabungkan WHERE dengan MAX & MIN

Penggunaan MAX dan MIN dapat dikombinasikan dengan WHERE, sama caranya dengan function agregat lainnya. Pada contoh di bawah ini, kita menggunakan MAX dan WHERE untuk mendapatkan pembelian paling mahal yang dilakukan oleh Ninja Ken.

```
sample.sql
SELECT MAX(price) ② Dapatkan nilai maksimum dari hasil yang telah ditemukan
FROM purchases
WHERE character_name = "Ninja Ken";
① Cari baris dengan nilai "Ninja Ken"
```

**Hubungi kami!**

Mengelompokkan Baris

Total jumlah dari setiap tanggal

SUM(price)	purchased_at
4	2018-10-10
3	2018-10-11
18	2018-10-12

**GROUP BY**

Dengan GROUP BY, Anda dapat mengelompokkan baris. Misalnya, menggunakan syntax GROUP BY column\_name, sebagaimana ditampilkan di bawah ini, baris dengan nilai yang sama akan dielompokkan untuk kolom yang ditunjukkan.

```
sample.sql
GROUP BY column_name
```

Hasil
-------

Hasil: purchases

ID	Name	Price	purchased_at
1	puding	1	2018-10-10
2	pena	1	2018-10-10
3	buku tulis	2	2018-10-10
4	puding	2	2018-10-11
5	teh	2	2018-10-11
6	rantangan	5	2018-10-11
7	teks	1	2018-10-12
8	teks	1	2018-10-12
9	teks	1	2018-10-12

Hasil: grup "2018-10-10"

Hasil: grup "2018-10-11"

Hasil: grup "2018-10-12"

Mengelompokkan & Mengagregat

Untuk mengelompokkan setiap baris dengan data agregat, Anda dapat menggunakan GROUP BY! column\_name pada akhir statement SQL, seperti yang ditunjukkan di bawah ini. Baris dengan nilai yang sama akan dielompokkan ke dalam satu baris.

```
sample.sql
GROUP BY purchased_at;
```

**Catatan Tentang GROUP BY**

Bagi pengguna awam, GROUP BY belum lagi cukup untuk mengelompokkan item atau fungsi agregat semisal dengan didalam didalam SELECT. Sebenarnya, di bawah ini, karena fungsi agregat SUM() tidak dilakukan pada kolom yang dilakukan oleh GROUP BY dan pada akhirnya yang dilengkapi adalah instance untuk setiap nilai purchased\_at.

**I-Grup**

```
sample.sql +
SELECT SUM(price)
FROM purchases
GROUP BY purchased_at;
```

**II-Aggregat**

Sum(price)	purchased_at
4	2018-10-10
4	2018-10-11
18	2018-10-12

**sample.sql +**

```
SELECT SUM(price), purchased_at
FROM purchases
GROUP BY purchased_at;
```

**sample.sql +**

```
SELECT price, purchased_at
FROM purchases
GROUP BY purchased_at;
```

Mengelompokkan Hasil dari Banyak Kolom

Jumlah total pengeluaran per tanggal dan nama (contoh)

Sum(price)	purchased_at	character_name
68	2018-10-10	Birdie
83	2018-10-10	Ninja Ken
279	2018-10-10	Guru Domba
8	2018-10-11	Baby Ben
35	2018-10-11	Ninja Ken
50	2018-10-12	Ninja Ken

Cara Menggunakan GROUP BY dengan banyak Kolom

Anda dapat menggunakan GROUP BY untuk banyak kolom dengan memasukkan nama kolom tersebut di dalamnya dengan koma (,). Kita akan lihat bagaimana pengelompokan dan agregat yang dapat dibuat di bawah berikutnya.

**sample.sql +**

```
GROUP BY kolom1,kolom2, gunakan koma
```

**sample.sql +**

```
SELECT SUM(price), purchased_at, character_name
FROM purchases
GROUP BY purchased_at, character_name;
```

GROUP BY dengan Banyak Kolom

Mengurangkan GROUP BY dengan beberapa kolom dapat mengoptimalkan data untuk menghasilkan grup seperti di bawah ini.

grup '2018-10-10' untuk "Ninja Ken"

Subtotal	purchased_at	character_name
68	2018-10-10	Ninja Ken
83	2018-10-10	Ninja Ken

grup '2018-10-11' untuk "Guru Domba"

Subtotal	purchased_at	character_name
279	2018-10-11	Guru Domba
35	2018-10-11	Guru Domba

grup '2018-10-12' untuk "Baby Ben"

Subtotal	purchased_at	character_name
8	2018-10-12	Baby Ben

Hasil Penggunaan GROUP BY dengan banyak Kolom

Menggunakan function agregat pada grup tertentu akan lebih efektif daripada menggunakan pada semua hasil. Dibawah ini adalah contoh penggunaan function SUM dan COUNT pada hasil yang telah diperbaiki.

total per tanggal dan nama karakter

SubTotal	purchased_at	character_name
50	2018-10-10	Ninja Ken
150	2018-10-10	Guru Domba
2	2018-10-10	Baby Ben
10	2018-10-11	Ninja Ken
27	2018-10-11	Guru Domba
9	2018-10-12	Ninja Ken

Jumlah pembelian per tanggal dan nama karakter

Count()	purchased_at	character_name
2	2018-10-10	Ninja Ken
2	2018-10-10	Guru Domba
1	2018-10-10	Baby Ben
3	2018-10-11	Ninja Ken
1	2018-10-11	Guru Domba
1	2018-10-12	Ninja Ken

Mengelompokkan Menurut Kondisi Tepat

Jumlah total pengeluaran untuk makanan per tanggal dan nama karakter

Sum(price)	purchased_at	character_name
29	2018-10-10	Birdie
32	2018-10-10	Ninja Ken
71	2018-10-10	Guru Domba
8	2018-10-11	Baby Ben
22	2018-10-11	Ninja Ken
36	2018-10-13	Ninja Ken

WHERE & GROUP BY

GROUP BY juga dapat digunakan dengan klausula WHERE. Untuk mendukung hal ini, fungsi GROUP BY seharusnya dikenakan setelah WHERE. Selain itu, fungsi COUNT, SUM, AVG, MAX, dan MIN juga dapat digunakan setelah WHERE.

**sample.sql +**

```
SELECT aggregate_function
FROM table_name
WHERE kondisi
GROUP BY kolom1, kolom2;
diikuti setelah WHERE!
```

Pencarian	WHERE
Gruping	GROUP BY
Function	COUNT - SUM - AVG - MAX - MIN

Urutan Penggunaan WHERE dan GROUP BY (1)

Untuk mendasarkan jumlah total uang makanan yang dibelanjakan setiap karakter pada hari tertentu, ikuti 3 langkah berikut:

- ① Carilah dengan nilai category "makanan"
- ② Kelompokkan nilai menurut kolom character\_name dan purchased\_at. Untuk langkah ketiga akan...

**1 Pencarian**

price	purchased_at	character_name	category
8	2018-10-10	Baby Ben	makanan
19	2018-10-10	Birdie	makanan
9	2018-10-10	Ninja Ken	makanan
47	2018-10-10	Ninja Ken	makanan
35	2018-10-10	Guru Domba	makanan
122	2018-10-10	Guru Domba	makanan

**2 Gruping**

Sum(price)	purchased_at	character_name
29	2018-10-10	Birdie
32	2018-10-10	Ninja Ken
71	2018-10-10	Guru Domba
8	2018-10-11	Baby Ben
22	2018-10-11	Ninja Ken
36	2018-10-13	Ninja Ken

Urutan Penggunaan WHERE dan GROUP BY (2)

Terakhir! Terapkan function agregat untuk mengelompokkan hasil.

**3 Aggregate**

Sum(price)	purchased_at	character_name
300	2018-10-10	Baby Ben
1000	2018-10-10	Ninja Ken
350	2018-10-10	Guru Domba
700	2018-10-11	Ninja Ken
270	2018-10-11	Guru Domba
500	2018-10-12	Ninja Ken

Menulis SQL dengan WHERE dan GROUP BY

**sample.sql +**

```
SELECT SUM(price), purchased_at, character_name
FROM purchases
WHERE category = 'makanan'
AND character_name = 'makanan'
GROUP BY purchased_at, character_name;
2.Grup hasil berdasarkan fungsi dan nama karakter!
```

Mempersempit Data yang Dikelompokkan

Tabel setelah gruppig

Sum(price)	purchased_at
4	2018-10-10
3	2018-10-11
18	2018-10-12

Dapatkan baris yang sudah di grup dengan jumlah total lebih besar dari 10

HAVING

Jika Anda ingin mempersempit pengelompokan hasil data yang dikelompokkan menggunakan GROUP BY, Anda dapat menggunakan HAVING. Sebagaimana ditampilkan pada contoh di bawah, kita bisa mendapatkan grup dengan kondisi tertentu dengan menggunakan kuintian GROUP BY column\_name HAVING kondisi.

**sample.sql +**

```
GROUP BY column_name
HAVING kondisi;
```

**Tabel setelah gruppig**

Sum(price)	purchased_at
4	2018-10-10
4	2018-10-11
18	2018-10-12

Hanya group ini yang memenuhi kondisi

WHERE & HAVING

Untuk mempersempit data setelah pengelompokan, gunakanlah HAVING, bukan WHERE, karena SQL memiliki kiat specific dalam melaksanakan kueriinya. Sebagaimana ditampilkan pada gambar di bawah, WHERE dijalankan pertama, lalu GROUP BY, lalu function, dan HAVING dijalankan terakhir.

Pencarian	WHERE
Gruping	GROUP BY
Function	COUNT - SUM - AVG - MAX - MIN
HAVING	HAVING

Perbedaan Antara

Bergantung pada urutannya, WHERE dan HAVING mencari target yang berbeda. WHERE mencari seluruh entitas dalam tabel dan mempersempitnya berdasarkan kriteria yang ditentukan. Sedangkan HAVING mempersempit hasil yang dikelompokkan.

Catatan Tentang

Karena HAVING mencari dari tabel setelah pengelompokan, kolom yang digunakan dalam permintaan

## WHERE dan HAVING

WHERE
GROUP BY
COUNT - SUM - AVG - MAX - MIN
HAVING

WHERE
id
name
price
1
puding
2
pene
3
bukuk tulis
4
puding
2

Mencari keseluruhan tabel

HAVING
SUM(price)
date
4
2018-10-10
4
2018-10-11
18
2018-10-12

Hanya mencari baris yang telah di grup

## HAVING

```
sample.sql = 
SELECT SUM(price), purchases_at
FROM purchases
GROUP BY purchased_at
HAVING SUM(price) > 10;
Menggunakan bilangan setelah gruping
```

Tabel setelah gruping	
SUM(price)	purchased_at
4	2018-10-1
4	2018-10-11
18	2018-10-12

Hanya group ini yang memenuhi kondisi

## Mempraktikkan Semua Yang Sudah Anda Pelajari



## Tabel Baru



## Tabel players



players

Nama Kolom	Data Tersimpan
id	nomor ID
name	nama player
goals	skor
height	tinggi

Hubungi kami!

## Menggunakan 2 Statement SQL



## Subkueri

Dalam SQL, Anda dapat menulis statement dalam statement, statement ini dinamakan subkueri. Hal ini memungkinkan Anda menggunakan beberapa statement SQL dalam 1 sehingga bisa mendapatkan data yang lebih kompleks. Contoh di bawah ini menunjukkan cara mendapatkan nama pemain dengan skor yang lebih tinggi dari Will.

```
SQL study I & II
sample1.sql =
SELECT goals
FROM players
① deapatkan skor Will
WHERE name = "Will";
sample2.sql =
SELECT name
FROM players
WHERE goals > ② sebelumnya, masukan skor will
```

```
Menggunakan subkueri
sample.sql =
SELECT name
FROM players
WHERE goals > ① subkueri
SELECT goals
FROM players
WHERE name = "Will"
);
```

Hubungi kami!

## Menulis Subkueri

Anda dapat membuat subkueri dengan menulis kembali dalam tanda kurung seperti di bawah ini. Menulis subkueri umumnya sama seperti menulis statement SQL biasa, subkueri tidak memerlukan tanda kurung.

Hanya satu tanda kurung yang diperlukan di akhir statement SQL.

```
sample.sql =
SELECT name
FROM players
WHERE goals > ( ① letakan subkueri di dalam ()
SELECT goals
FROM players
WHERE name = "Will"
);
);
```

## Alur Menjalankan Code

Jika statement SQL mencakup subkueri, code subkueri hanya akan dijalankan terlebih dahulu sebelum bagian lain statement SQL-nya, seperti contoh di bawah ini.

```
sample.sql =
SELECT name
FROM players
WHERE goals = ① deapatkan skor Will
SELECT goals
FROM players
WHERE name = "Will"
);
```

```
sample.sql =
② deapatkan name pemain dengan skor lebih tinggi dari skor Will
SELECT name
FROM players
dapatkan dan
③ gunakan skor
SELECT goals
FROM players
WHERE name = "Will"
);
```

Hubungi kami!

## Membuat Data Lebih Mudah Dibaca



## Menggunakan AS

Dengan menggunakan AS, Anda dapat membuat label sementara untuk tabel dan nama kolom. Untuk melakukannya, gunakan sintaks column\_name AS "Nama Baru". Ini akan menampilkan column\_name sebagai Nama Baru.

```
sample.sql =
SELECT goals AS "Skor Will"
FROM players
④ mengambilkan cara hasil dari kolom
goal di kompilkan
WHERE name = "Will";
```



14

## Tabel countries

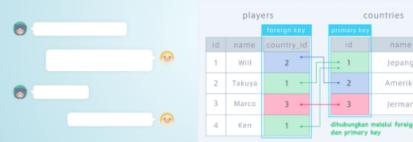
countries	
Nama Kolom	Data Tersimpan
id	Nomor ID
name	Nama Negara
rank	Peringkat Tim

## Menggabungkan Tabel (1)



✉ Hubungi kami!

## Menggabungkan Tabel (2)



## Manfaat Menggabungkan Tabel

Tanpa menggunakan banyak tabel			
1	Will	Amerika	24
2	Takuya	Jepang	55
3	Marco	Jerman	1
4	Ken	Jepang	55

membutuhkan perubahan di dua baris ketika peringkat berubah

menggunakan beberapa tabel			
1	Will	2	1
2	Takuya	1	2
3	Marco	3	3
4	Ken	1	4

hanya membutuhkan 1 perubahan

## Apa itu JOIN?

JOIN digunakan untuk menggabungkan beberapa tabel menjadi satu. Kondisi untuk penggabungan ditentukan dengan ON. Setelahnya Anda bisa mendapatkan data dari tabel hasil penggabungan selainnya dari satu tabel. Di bawah ini adalah contoh cara menggabungkan dua tabel, tabel\_a dan tabel\_b.

```
sample.sql >
SELECT *
FROM tabel_a
JOIN tabel_b
ON kondisi
```

JOIN (digabungkan)			
1	Will	2	1
2	Takuya	1	2

## Perintah Eksekusi untuk JOIN

Sejauh yang ditampilkan contoh dibawah ini, jika statement SQL berisi klausus JOIN, JOIN akan dijalankan terlebih dahulu. Berikutnya, SELECT akan dijalankan di tabel gabungan.

```
sample.sql >
SELECT *
FROM players
JOIN countries
ON players.country_id = countries.id;
```

② selesainya, kueri SELECT akan di jalankan di hasil dari kueri JOIN			
1	Will	2	Amerika
2	Takuya	1	Jepang

## Menentukan Kolumn Bernama Sama di Beberapa Tabel (1)

Jika terdapat nama kolom yang sama di lebih dari satu tabel, nama tersebut harus ditentukan dengan `table_name.column_name`. Di bawah ini adalah contoh kolom name yang terdapat di beberapa tabel.

```
sample.sql >
SELECT name, name
FROM players
JOIN countries
ON players.country_id = countries.id;
```

Kolomnya dapat di spesifikasi dengan menentukan nama tabel			
1	players.name	countries.name	

## Alur Pengerjaan Code

Cara kerja alur pengerjaan statement SQL. Perintah bahwa FROM & JOIN ditarikkan diawal karena table harus ditentukan sebelum penarikan SQL dapat dipakai.



## Tabel teams (2)

players			
1	Will	—	1
2	Takuya	—	1
3	Marco	—	2
4	Takuya	—	2
5	Michael	—	2

NULL

## Cara Penggabungan Tabel

DI SQL, foreign key dan primary key digunakan untuk menggabungkan tabel. Dengan key di tabel lain dengan foreign key, Anda dapat menghubungkan tabel.

players			
1	Will	2	1
2	Takuya	1	2
3	Marco	3	3
4	Ken	1	4

dihitung melalui foreign key dan primary key

## Menggabungkan Tabel

✉ Hubungi kami!

## Kondisi Penggabungan

Untuk mensusulkan kondisi penggabungan, gunakan sintaks `ON table_a.column_name = table_b.column_name`. Dalam contoh di bawah bagian kolan, country\_id di tabel `players` dan di tabel `countries` digunakan untuk penggabungan.

```
sample.sql >
SELECT *
FROM tabel_a
JOIN tabel_b
ON tabel_a.column_name = tabel_b.column_name
```

```
sample.sql >
SELECT *
FROM players
JOIN countries
ON players.country_id = countries.id;
```

## Mendapatkan Data dengan JOIN

✉ Hubungi kami!

## Menentukan Kolumn Bernama Sama di Beberapa Tabel (2)

Syntax `table_name.column_name` juga bisa ada di clause WHERE. Saat menangan beberapa tabel, penting untuk diingat bahwa mungkin terdapat nama kolom yang sama di tabel yang berbeda.

```
sample.sql >
SELECT *
FROM players
JOIN countries
ON players.country_id = countries.id
WHERE players.name = "Will";
```

✉ Hubungi kami!

## Tabel teams (1)

teams	
Nama Kolom	Data Tersimpan
id	... Nomor ID
name	... Nama Tim

Saat menggunakan JOIN, pengertian code dijalankan berdasarkan tabel yang ditentukan dengan `FROM`. Namun, baris dengan nilai `foreign key` NULL, seperti yang ditampilkan contoh di bawah ini, tidak akan ditampilkan di hasil yang dijalankan.

```
sample.sql >
SELECT *
FROM players
JOIN teams
ON players.previous_team_id = teams.id
```

players			
1	Will	—	1
2	Takuya	—	1
3	Marco	—	2
4	Takuya	—	2
5	Michael	—	2

✉ Hubungi kami!

## Perbandingan dengan NULL



## Menggabungkan 3 Tabel atau Lebih



## Mempraktikkan Hal Yang Sudah Anda Pelajari



## Menambahkan Data



## AUTO INCREMENT

Dalam banyak kasus, kolom id di SQL akan menggunakan fitur AUTO\_INCREMENT. Ketika data baru ditambahkan ke dalam tabel, AUTO\_INCREMENT akan menambahkan id data baru tersebut secara otomatis.

```
exercise.sql
INSERT INTO students (name, course)
VALUES ('Kate', 'Java');
Kolom AUTO_INCREMENT tidak diperlukan
```

AUTO_INCREMENT		
id	name	course
2	Emma	SQL
3	Chris	HTML
4	kate	Java

## UPDATE

Klaus UPDATE digunakan ketika Anda ingin memperbarui data. Jika Anda ingin memperbarui data di beberapa kolom, gunakan koma (,) untuk memisahkan setiap nama kolomnya.

```
exercise.sql
UPDATE students
SET name = 'Jordan', course = 'HTML'
WHERE id = 6;
perbarui nilai dari kolom yang sudah diperbarui
specifikasi record yang ingin diperbarui menggunakan WHERE
```

students		
id	name	course
5	Tom	SQL
6	Jordan	HTML

## Sebelum Menjalankan UPDATE

Data selalu harus diolah sebelum menjalankan kueri UPDATE. Karena itu, Anda harus membuatkan untuk menjalankan SELECT setelah sebelumnya guna mengonfirmasi data yang akan proses.

Langkah 1: Periksa data sebelum memprosesnya		
id	name	course
6	Jaden	PHP

Langkah 2: Proses data yang sudah dikonfirmasi		
id	name	course
6	Jordan	HTML

## LEFT JOIN

Menunjukkan LEFT JOIN. Anda bisa mendapatkan semua data tabel yang ditentukan di klausus FROM. Baris dengan nilai foreign key NULL juga akan ditampilkan di hasil yang dijalankan, sebagai NULL.

id	name	previous_team_id	id	name
1	Will		1	Ninja Warriors
2	Takuya	1	1	Ninja Warriors
3	Marco		2	Takuya
4	Takuya		3	Marco
5	Michael	2	2	Red Riders
6	Sho	4	4	White Rams

## JOIN dengan Beberapa Tabel

JOIN adalah statement SQL tunggal yang dapat digunakan lebih dari sekali. Perhatikan bahwa, meskipun Anda menggunakan JOIN lebih dari satu kali, Anda hanya perlu menulis FROM satu kali.

sample.sql
<pre>SELECT * FROM players JOIN countries ON players.country_id = countries.id LEFT JOIN teams ON players.previous_team_id = teams.id;</pre>

✉ Hubungi kami!

## Pengolahan Data SQL

Dalam pelajaran ini, kita akan mempelajari sintaks yang digunakan untuk memambah, memperbarui, dan menghapus data.

Metode pengolahan data ini digunakan dalam semua layanan web, seperti toko online dan media sosial.



## INSERT

Klausus INSERT digunakan ketika menambahkan catatan baru ke dalam tabel.

students		
id	name	course
2	Emma	SQL
3	Chris	HTML

students		
id	name	course
4	kate	Java

✉ Hubungi kami!

## Ketika Anda Memasukkan Data yang Salah...

students		
id	name	course
5	Tom	SQL
6	Jordan	HTML
7	Emily	PHP

students		
id	name	course
5	Tom	SQL
6	Jordan	HTML
7	Emily	PHP

Saat ingin memperbaiki data yang salah

## Point Penting untuk UPDATE

Poin Penting untuk UPDATE

Jika Anda ingin memperbarui data yang ingin Anda perbarui menggunakan klausus WHERE secara spesifik, maka semua data di kolom akan diperbarui.

students		
id	name	course
5	Jordan	HTML
6	Jordan	HTML
7	Jordan	HTML

students		
id	name	course
5	Jordan	HTML
6	Jordan	HTML
7	Jordan	HTML

✉ Hubungi kami!

## Sebelum Menjalankan UPDATE

Data selalu harus diolah sebelum menjalankan kueri UPDATE. Karena itu, Anda harus membuatkan untuk menjalankan SELECT setelah sebelumnya guna mengonfirmasi data yang akan proses.

Langkah 1: Periksa data sebelum memprosesnya		
id	name	course
6	Jaden	PHP

Langkah 2: Proses data yang sudah dikonfirmasi		
id	name	course
6	Jordan	HTML

## Bagaimana dengan Data yang Tidak Diperlukan

students		
id	name	course
6	Jordan	HTML
7	Emily	PHP

7	Emily	PHP
---	-------	-----

Gunakan SELECT untuk memeriksa apakah Anda memproses data yang benar.

7	Emily	PHP
---	-------	-----

⚠ Data tidak dapat diambil sebelum code dijalankan

8	Julia	SQL
---	-------	-----

Saat ingin menghapus data yang tidak diperlukan

### DELETE

Klausus DELETE digunakan untuk menghapus data. Sama seperti UPDATE, data yang telah dihapus tidak dapat dipulihkan kembali, jadi manfaatkanlah data yang ingin kita hapus terlebih dahulu dengan menggunakan SELECT.

```
exercise.sql >
DELETE FROM students
WHERE id = 7;
```

Spesifikasi data yang ingin dihapus menggunakan WHERE

id	name	course
6	Jordan	HTML
7	Emily	PHP
8	Julia	SQL

### Poin Penting untuk DELETE

Berhati-hatilah! Jika Anda tidak menentukan data yang ingin Anda hapus secara spesifik dengan menggunakan WHERE, semua data Anda di dalam tabel tersebut akan dihapus.

```
exercise.sql >
DELETE FROM students
```

Belum dispesifikasi menggunakan WHERE

id	name	course
6	Jordan	HTML
7	Kate	PHP
8	Julia	SQL

### Meninjau Empat Klausu

### Empat Statement

Ayo tinjau keempat klausu yang baru kita pelajari dalam pelajaran ini. Ketika menjalankan klausu UPDATE dan DELETE, Anda harus selalu menggunakan WHERE untuk menspesifikasi data yang akan diproses.

Mengambil Data

```
SELECT columnA, columnB  
FROM tableName;
```

Memperbarui Data

```
UPDATE tableName  
SET columnA = nilai1, columnB = nilai2  
WHERE kondisi;
```

Selalu spesifikasi

Menambahkan Data

```
INSERT INTO tableName (columnA, columnB...)  
VALUES (nilai1, nilai2...);
```

Menghapus Data

```
DELETE FROM tableName  
WHERE kondisi;
```

Selalu spesifikasi

[✉ Hubungi kami!](#)