diskominfo jabar | JABAR DIGITAL SERVICE | JABAR DIGITAL ACADEMY | aws InCommunities

MySQL QUERY DML

KELAS PROGRAMMING FULL STACK DEVELOPER

MITRA PELATIHAN

Alkademi

2024

Jabar Digital Academy

digitalacademy.jabarprov.go.id

BAB V
MYSQL DML

A. Tujuan.

a. Peserta didik dapat memahami apa itu DML.
b. Peserta didik dapat memahami penggunaan DML.
c. Peserta didik dapat memahami konsep CRUD dalam pengelolaan database.
d. Peserta didik dapat memahami cara *insert* data kedalam tabel.
e. Peserta didik dapat memahami penggunaan query *SELECT*.
f. Peserta didik dapat memahami penggunaan *Alias* dalam query SELECT.
g. Peserta didik dapat memahami penggunaan *Where* dalam query SELECT.
h. Peserta didik dapat memahami penggunaan ORDER BY dalam query SELECT.
i. Peserta didik dapat memahami penggunaan Limit dalam query SELECT.
j. Peserta didik dapat memahami penggunaan Distinct dalam query SELECT.
k. Peserta didik dapat memahami penggunaan Group By dalam query SELECT.
l. Peserta didik dapat memahami penggunaan Having dalam query SELECT.
m. Peserta didik dapat memahami penggunaan Join dalam query SELECT.
n. Peserta didik dapat memahami penggunaan Subquery dalam query SELECT.
o. Peserta didik dapat memahami penggunaan Update dalam query SELECT.
p. Peserta didik dapat memahami penggunaan Delete dalam query SELECT.

B. Perlengkapan.

a. Modul Bab 4. MySQL DML.
b. Komputer dengan sistem operasi Windows bila diperlukan
c. Software Xampp.

C. Materi.

Setelah kita mempelajari DDL, bagaimana kita mengelola database dengan query yang ada dalam DDL, dengan memahami DML kita dapat mengetahui bagaimana cara mengelola data di dalamnya. Dengan menguasai DML, kita dapat melakukan berbagai operasi penting seperti memasukkan data baru, mengubah data yang sudah ada, menghapus data yang tidak diperlukan, dan mengambil data yang diperlukan. Kemampuan untuk menggunakan DML dengan baik memungkinkan pengguna

untuk efektif dan efisien mengelola dan mengontrol data dalam database, yang merupakan hal krusial dalam berbagai aplikasi dan sistem informasi.

5) Apa Itu DML

DML adalah sebuah struktur. Pada bagian ini kita akan mulai mempelajari cara mengolah data pada *database*. Kita akan mempelajari prosedur yang biasa dikenal dengan CRUD (*Create*, *Read*, *Update*, dan *Delete*). Mari kita pelajari satu per-satu fungsi-fungsi tersebut.

a. Create (Insert)

Insert adalah salah satu perintah dalam sql yang fungsinya untuk menambah *record* maupun *field*. Kita perlu didefinisikan terlebih dahulu tabel dan kolomnya jika ada kolom yang tidak terdefinisikan saat menambahkan data maka nilai dari kolom tersebut akan menjadi null atau kosong. Query *Insert* data kedalam tabel seperti dibawah ini.

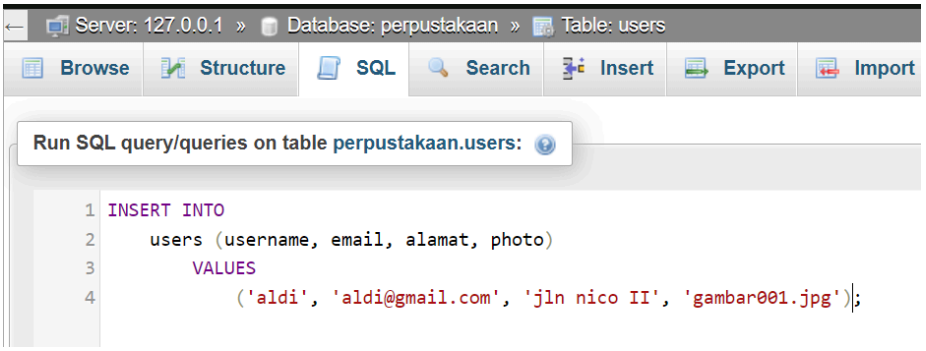
```
INSERT INTO
    (nama_tabel) (kolom1, kolom2)
    VALUES
        (value1, value2, value3);
```

Kita juga dapat menambahkan banyak data dalam satu kali perintah, cukup tambahkan data yang akan disimpan setelah data awal lalu pada data awal beri koma. Contohnya seperti ini.

```
INSERT INTO
    (nama_tabel) (kolom1, kolom2)
    VALUES
        (value1, value2, value3),
        (value1, value2, value3),
        (value1, value2, value3);
```

Mari kita mulai untuk menambahkan data pada tabel users yang telah kita sebelumnya. Ikuti langkah berikut:

1. Buka menu “SQL” pada bagian atas
2. Tulis query di bawah ini



Query Insert

3. Tekan “Go”
4. Jika berhasil akan muncul pesan seperti ini

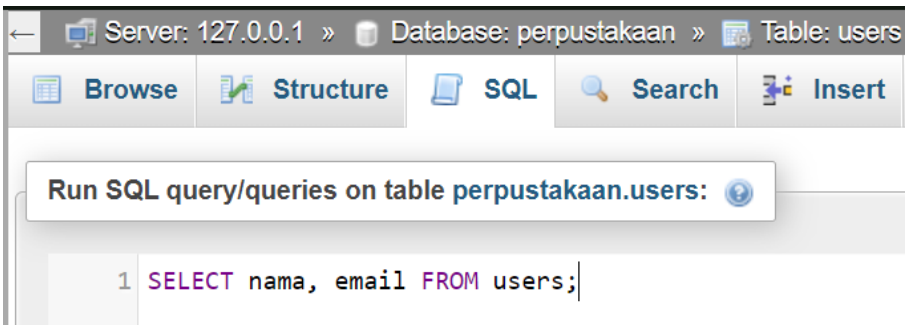
Respon Berhasil Insert Data

B. Read (Select)

Salah satu query dasar yang umum digunakan dalam MySQL adalah *SELECT*, yang berguna untuk menampilkan data dari tabel yang tersimpan dalam database. Contoh penggunaan query *SELECT* dapat dilihat di bawah ini:

```
SELECT (nama_column) FROM (nama_tabel);
```

Jadi setelah select kita tentukan nama column apa saja yang akan ditampilkan dari table yang sudah ditentukan. Contoh query untuk menampilkan tabel users.

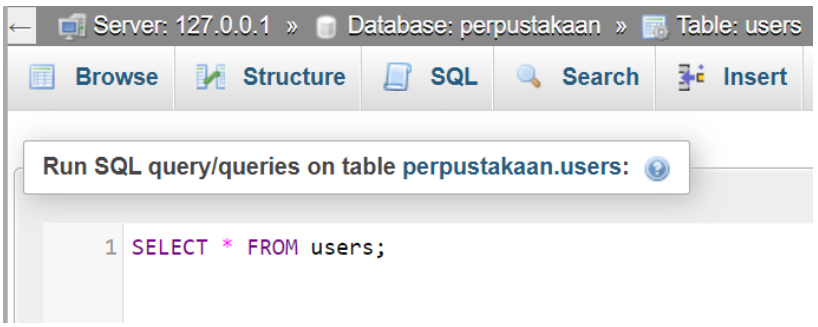


Query Menampilkan User.

Jika ingin menampilkan semua kolom dari tabel user cukup beri bintang sebelum *FROM*, seperti ini

```
SELECT * FROM (nama_tabel);
```

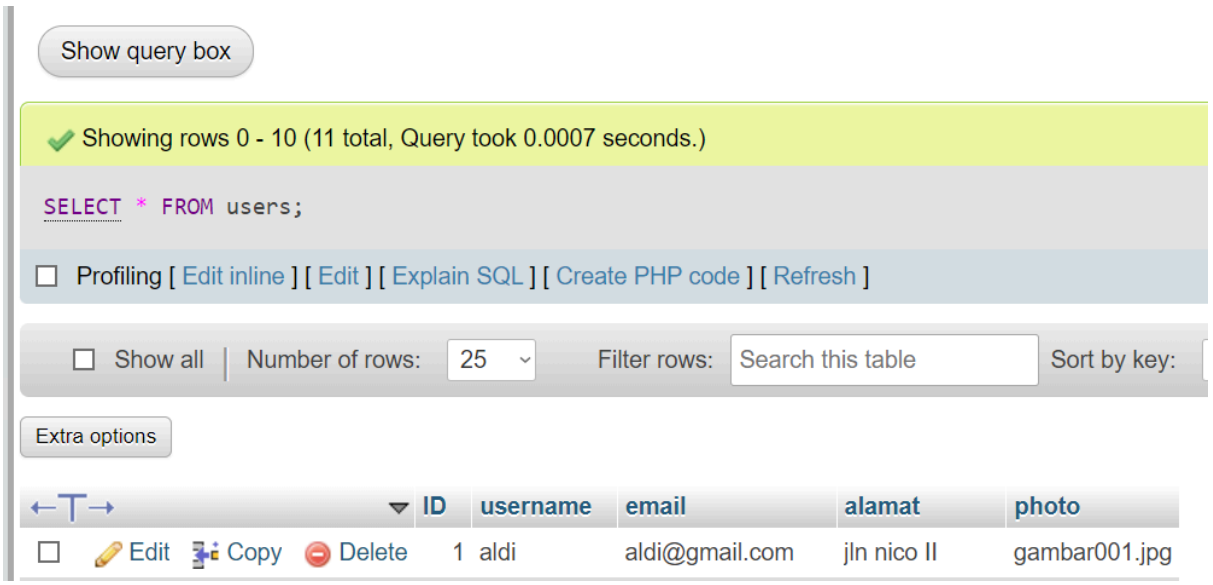
Kita langsung praktekan saja di PHPMyAdmin, kita coba tampilkan semua data yang ada di dalam tabel users.



Query Menampilkan Semua Data Didalam *Users*.

Dengan query *SELECT*, kita dapat memilih kolom-kolom tertentu yang ingin ditampilkan dari tabel yang ditentukan. Ini membantu kita dalam melihat dan memanipulasi data yang ada dalam database dengan mudah dan efisien.

Tekan “Go”, lalu apabila berhasil akan muncul tabel seperti berikut:



Hasil Query *Select*.

Kita sudah mencoba menggunakan query SELECT sederhana untuk menampilkan semua data dari tabel, ada banyak perintah yang dapat digunakan didalam query *SELECT* untuk mendukung kita agar dapat menampilkan data yang sesuai dengan yang kita inginkan. Misalnya ingin filter agar data yang tampil hanya beberapa saja, lalu ingin mengurutkan berdasarkan kolom tertentu. Dibawah ini beberapa perintah yang dapat dipakai di dalam query *SELECT* :

1. ALIAS

Saat kita ingin menampilkan data dari tabel tidak sedikit ada kolom yang namanya sama, kita dapat ubah nama kolom yang akan ditampilkan namun tidak merubah nilai di dalamnya, cara melakukannya yaitu dengan menggunakan Alias, Alias biasa digunakan untuk memberikan nama kolom agar mudah dibaca, contohnya misalnya CatId diubah menjadi CategoryId. Untuk membuat alias cukup berikan *AS* lalu diikuti dengan nama yang diinginkan, contohnya seperti ini :

```
SELECT (nama_kolom) AS (alias_name)
FROM (nama_tabel);
```

Kita langsung coba praktekan, ikuti langkah dibawah ini :

1. Pertama kita klik “SQL”
2. Lalu kita tambahkan data user, jalankan perintah Query seperti dibawah ini.

Run SQL query/queries on table perpustakaa

```
1 SELECT
2     username,
3     email,
4     alamat as address,
5     photo as image
6 FROM users;
```

Query *Alias* Pada Kolom.

3. Tekan "*Go*" maka data yang ditampilkan seperti ini.

Show query box

Showing rows 0 - 10 (11 total, Query took 0.0005 seconds.)

```
SELECT username, email, alamat as address, photo as image FROM users;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key:

Extra options

←T→				username	email	address	image
☐	Edit	Copy	Delete	aldi	aldi@gmail.com	jln nico II	gambar001.jpg

Hasil Dari Query *Alias*.

2. WHERE

Ketika kita ingin menampilkan data dari tabel menggunakan query *SELECT* kita dapat menambahkan filter agar data yang tampil sesuai dengan yang kita butuhkan. Query ini umum digunakan dalam MySQL yaitu *WHERE*, Penggunaan query Where disisipkan setelah query *SELECT* pada saat mengambil data, Jadi saat kita mengambil sebuah data menggunakan *SELECT*, kita dapat menggunakan perintah *WHERE* untuk memfilter baris data yang akan ditampilkan berdasarkan kondisi yang ditentukan. Penggunaan query ini cukup sederhana, tambahkan saja *WHERE* setelah nama tabel lalu diikuti dengan kondisi seperti ini :

```
SELECT (nama_kolom)
FROM (nama_tabel)
WHERE (nama_kolom) (operator) (nilai);
```

Kita dapat menambahkan lebih dari satu kondisi dalam satu query *WHERE* ada banyak jenis operator yang bisa digunakan didalam *WHERE*, cukup tambahkan *AND* untuk semua kondisi harus terpenuhi atau *OR* untuk salah satu dari kondisi terpenuhi, contoh penggunaannya seperti ini :

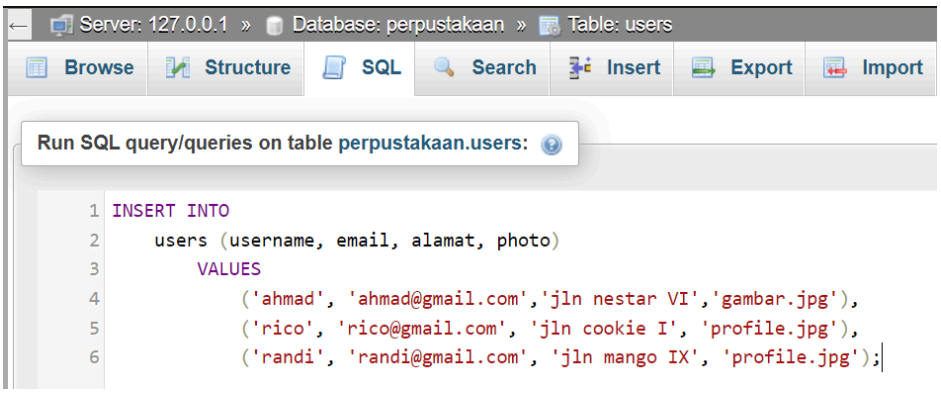
```
SELECT (nama_kolom)
FROM (nama_tabel)
WHERE (nama_kolom) (operator) (nilai)
(AND / OR) (nama_kolom) (operator) (nilai);
```

berikut jenis operator yang dapat digunakan :

Operator	Keterangan
=	Nilai dalam kolom yang sama dengan nilai yang ditentukan
>	Nilai dalam kolom yang lebih dari nilai yang ditentukan
<	Nilai dalam kolom yang kurang dari nilai yang ditentukan
>=	Nilai dalam kolom yang lebih dari sama dengan nilai yang ditentukan
<=	Nilai dalam kolom yang kurang dari sama dengan nilai yang ditentukan
<>	Nilai dalam kolom yang tidak sama dengan baik kurang maupun lebih nilai yang ditentukan, sebagai pengganti kita dapat menggunakan operator !=
BETWEEN	Nilai dalam kolom yang lebih dari sama dengan nilai yang ditentukan
LIKE	Data Binary Large Object besar (0-4294927695)
IN	Text ukuran sangat kecil (0-255)

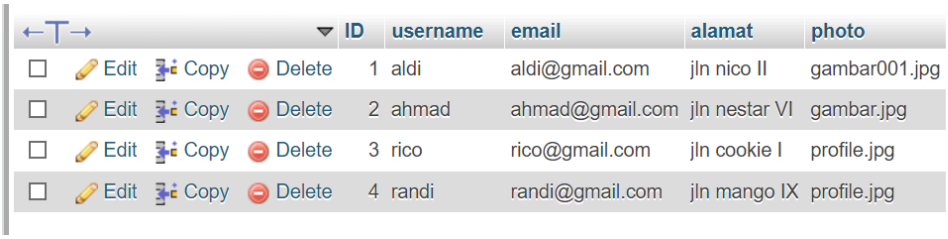
Kita langsung praktekan saja :

1. Pertama kita klik “SQL”
2. Lalu kita tambahkan data user, jalankan perintah Query seperti dibawah ini.



Query *Insert* Data.

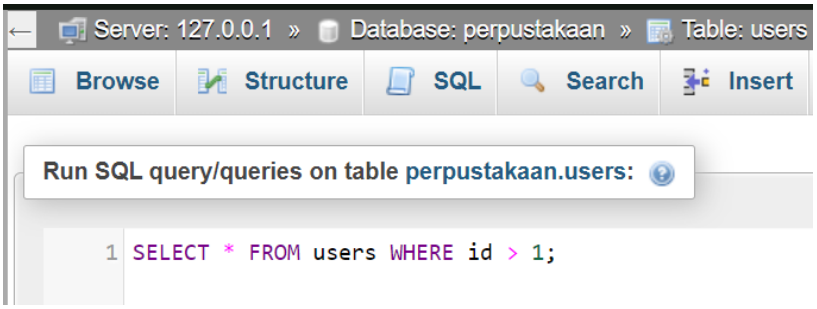
3. Lalu tekan *"Go"* dan pastikan data nya berhasil ditambahkan.



			ID	username	email	alamat	photo
Edit	Copy	Delete	1	aldi	aldi@gmail.com	jln nico II	gambar001.jpg
Edit	Copy	Delete	2	ahmad	ahmad@gmail.com	jln nestar VI	gambar.jpg
Edit	Copy	Delete	3	rico	rico@gmail.com	jln cookie I	profile.jpg
Edit	Copy	Delete	4	randi	randi@gmail.com	jln mango IX	profile.jpg

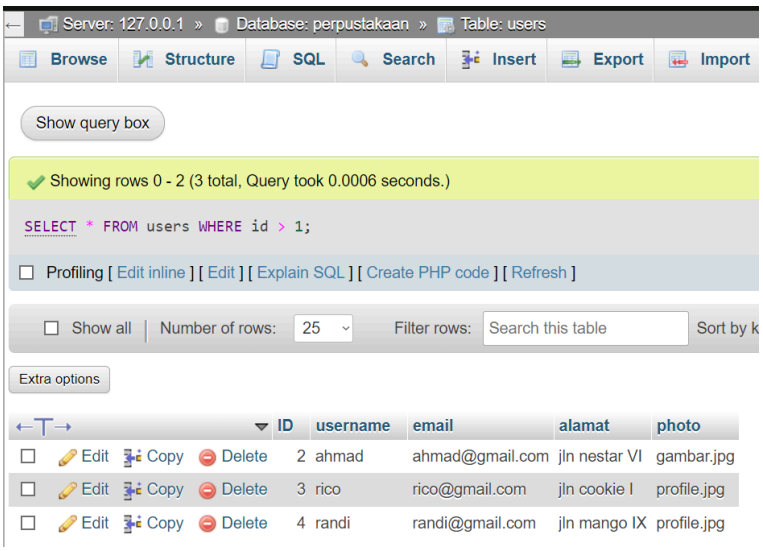
Tampilan Data Di Dalam Tabel Users.

4. Jika data didalam users sudah ditambahkan kita akan mencoba menampilkan namun *filter* dengan kondisi dimana data dari tabel users yang memiliki id lebih dari 1 maka tampilkan, perintah query nya seperti ini.



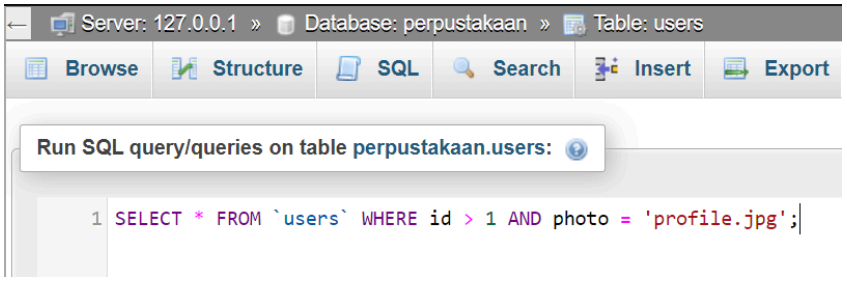
Query *Where*.

5. Jika sudah dijalankan maka akan menampilkan data user seperti dibawah ini.



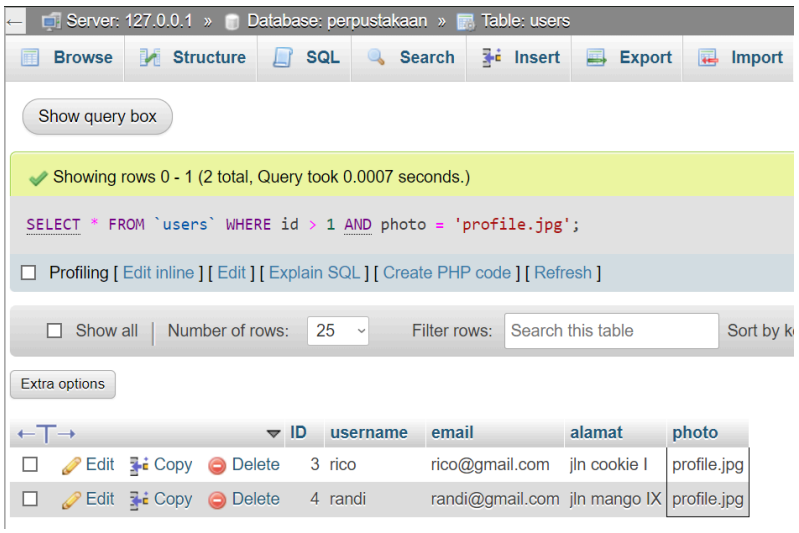
Response Berhasil Menampilkan Data.

6. Lalu kita coba tambahkan lagi satu kondisi dimana photo yang namanya "profile.jpg" dan jika kedua kondisi terpenuhi maka akan ditampilkan, maka kita cukup tambahkan *AND* setelah kondisi pertama seperti ini.



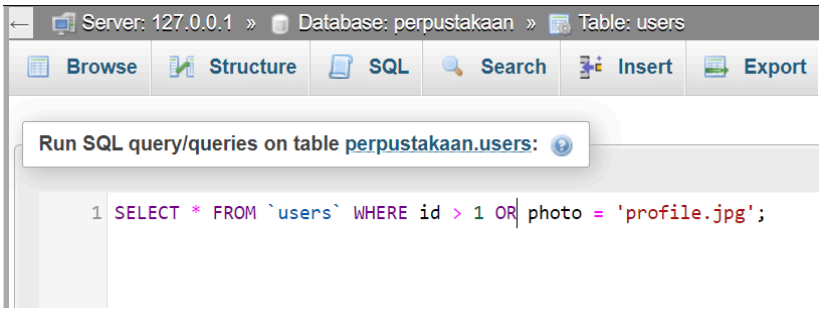
Query *Where* Yang Ditambahkan *AND*.

7. Tekan "*Go*" maka akan menampilkan data seperti ini.



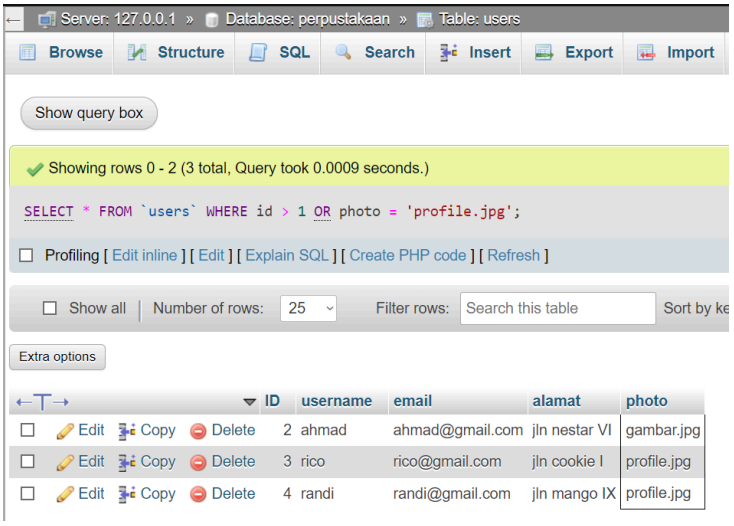
Tampilan data yang telah difilter.

8. Sekarang kita coba menggunakan *OR*, jika salah satu kondisi terpenuhi maka akan ditampilkan, perintah querynya seperti ini.



Query Where Ditambahkan *OR*.

9. Tekan "*Go*", maka data yang akan ditampilkan seperti ini.



Tampilan Data Setelah Difilter.

3. ORDER BY.

Ketika ingin menampilkan data menggunakan *SELECT* kita dapat mengurutkan urutan berdasarkan kolom, untuk mengurutkan data yang akan ditampilkan kita dapat menggunakan perintah *ORDER BY* ada 2 nilai yang dapat digunakan didalam perintah ini yaitu :

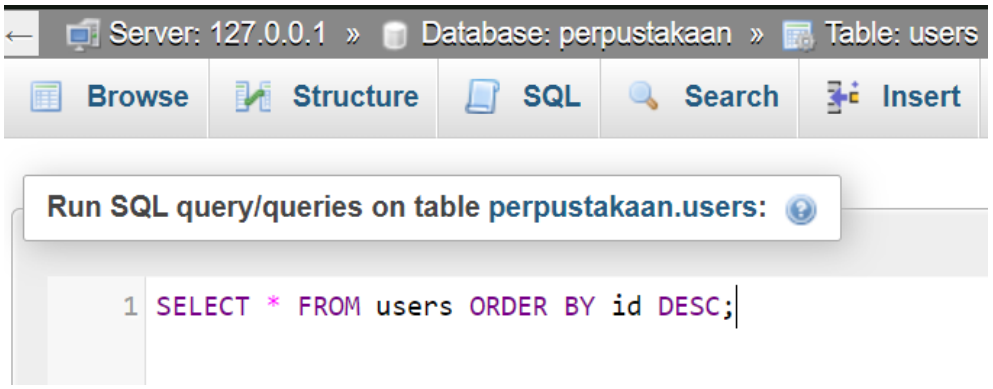
- ASC : Mengurutkan dari terkecil - terbesar.
- DESC : Mengurutkan dari terbesar - terkecil.

Contoh query ORDER BY seperti ini :

```
SELECT (nama_kolom, nama_kolom)
FROM (nama_tabel)
ORDER BY (nama_kolom, nama_kolom) (ASC / DESC);
```

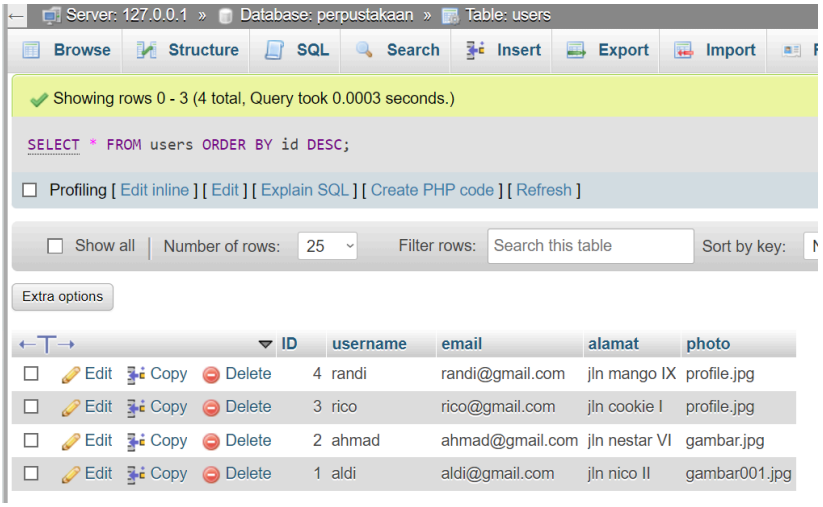
Kita langsung praktekan saja, ikuti langkah - langkah berikut ini:

1. Pertama kita klik “SQL”
2. Lalu kita coba tampilkan data yang ada di dalam tabel user dan urutkan berdasarkan ID dari yang terbesar hingga terkecil menggunakan DESC.



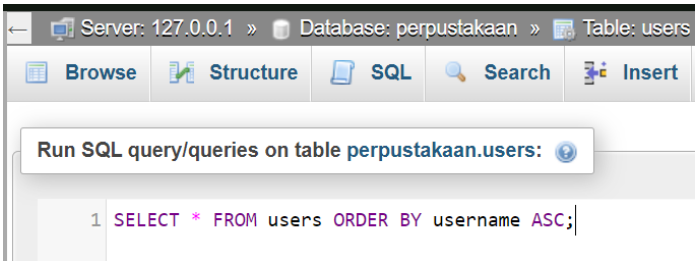
Query Order By Dengan Nilai DESC.

3. Jika sudah tekan “Go” maka akan menampilkan data dengan urutan dari ID terbesar hingga terkecil.



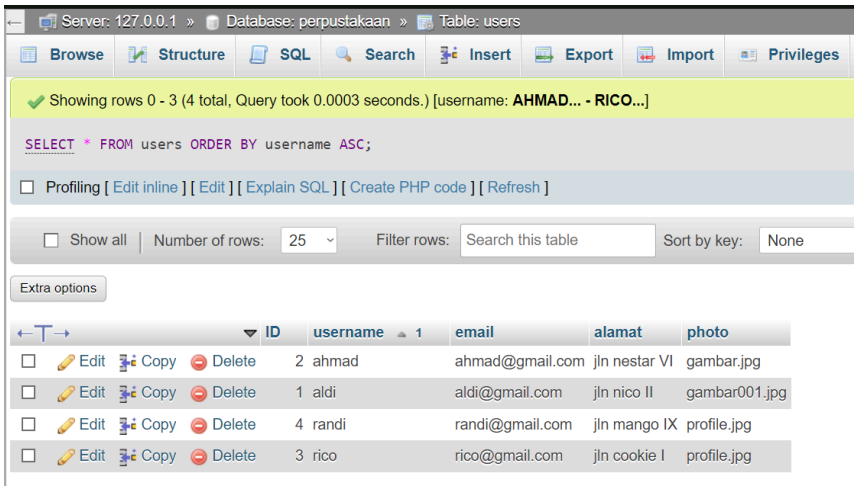
Tampilan Data Yang Telah Diurutkan.

4. Sekarang kita coba mengurutkan dari kolom username namun dari terkecil hingga terbesar menggunakan ASC.



Query Order By Dengan ASC.

5. Tekan "*Go*"



Tampilan Data Yang Telah Diurutkan.

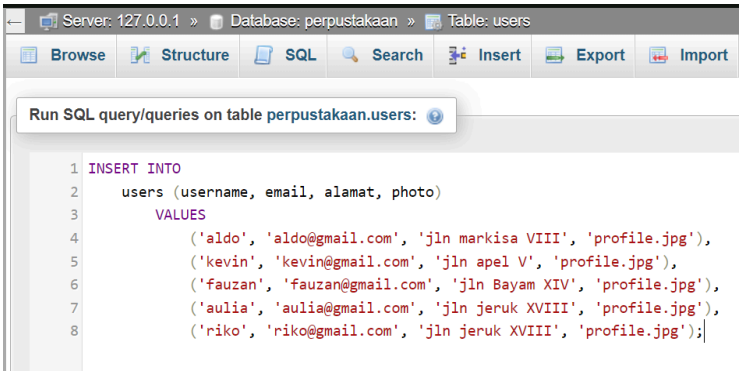
4. LIMIT

Data dalam tabel terkadang terlalu banyak data yang perlu diolah, dan lagi mengambil semua data yang banyak didalam tabel memerlukan waktu yang lama, untuk mengatasi itu kita perlu membatasi data yang akan ditampilkan menggunakan perintah *LIMIT*, dengan perintah ini kita dapat membatasi data yang akan ditampilkan sesuai dengan jumlah limit yang ditentukan. Untuk menambahkan limit di dalam query *SELECT* kita cukup tambahkan perintah *LIMIT* lalu diikuti dengan jumlah data yang akan dibatasi, contoh perintah limit seperti ini.

```
SELECT (nama_kolom, nama_kolom)
FROM (nama_table)
LIMIT (int);
```

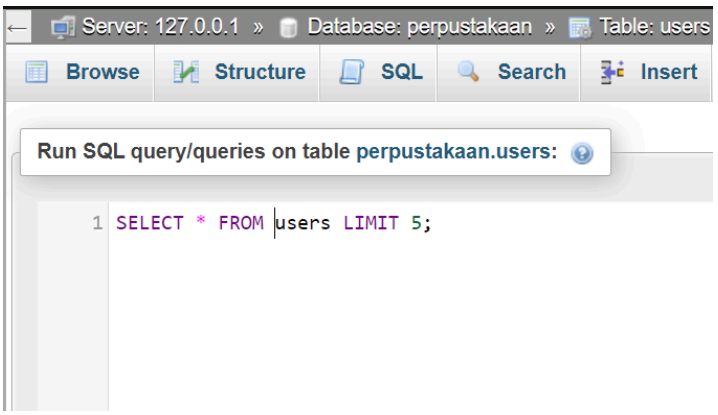
Kita langsung praktekkan saja, ikuti langkah - langkah dibawah ini :

1. Pertama kita kembali ke “SQL”
2. sebelum kita tampilkan semua data yang ada didalam tabel users, kita coba tambahkan lagi data yang ada di dalam tabel user, seperti dibawah ini.



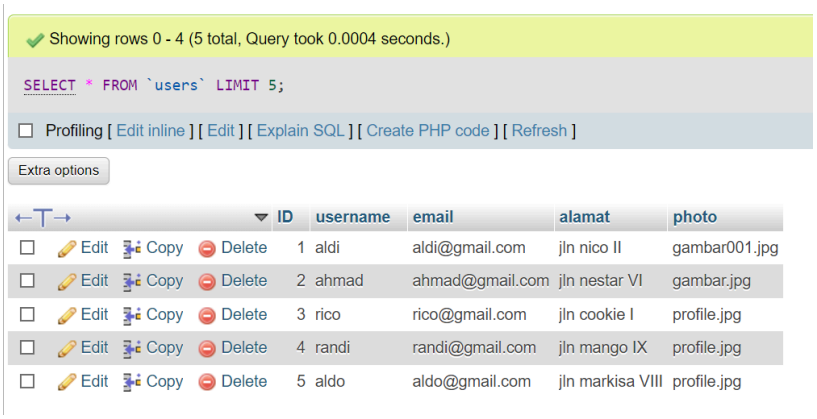
Query Insert Data.

3. Lalu kita coba untuk menampilkan semua data yang ada di dalam tabel user namun kita batasi menggunakan *LIMIT*, seperti ini.



Query *Limit*.

4. Lalu tekan "*Go*".



Tampilan Data Yang Sudah Dibatasi.

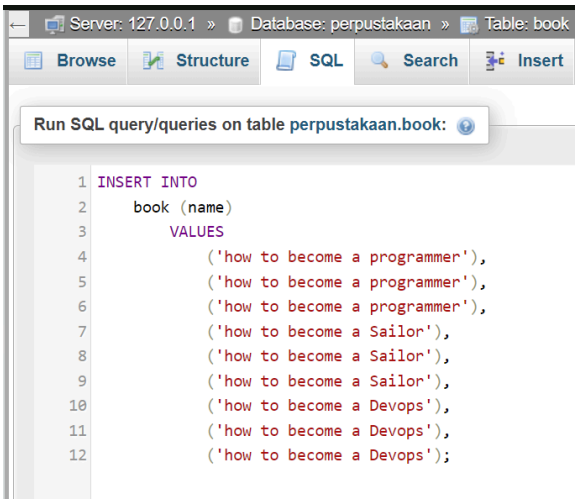
5. DISTINCT

Saat kita membuat query untuk menampilkan data pada suatu tabel, Sangat dimungkinkan kita akan mendapat duplikasi data pada baris tabel tersebut. Dengan menggunakan select distinct kita dapat menghapus duplikasi baris tersebut sehingga akan menampilkan data yang unik. Untuk membuat query distinct kita hanya perlu memanggil *Distinct* di baris dimana kita menentukan kolom mana saja yang akan di Berikut adalah contoh query Distinct :

```
SELECT DISTINCT
     (nama_kolom)
FROM (nama_tabel);
```

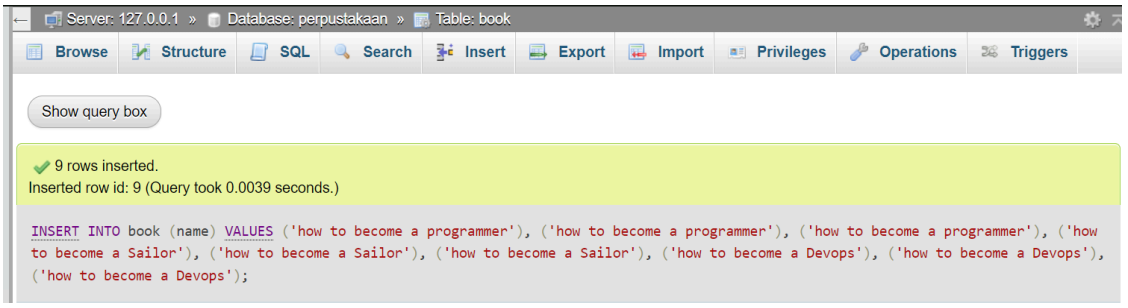
Kita langsung praktekan saja, ikuti langkah dibawah ini :

1. Pertama kita kembali ke “SQL”.
2. Kita perlu menambahkan lagi data namun dengan nilai yang sama, Sebelum kita mencoba Distinct dalam query kita, kita coba tambahkan data yang sama didalam tabel *book*.



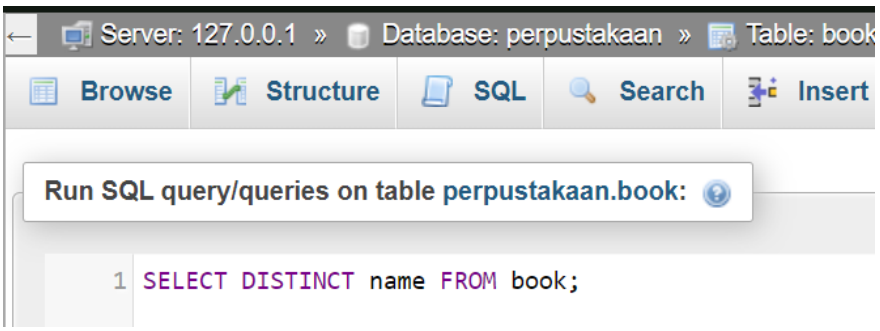
Query Menambahkan Data.

3. Tekan “*Go*” dan pastikan data sudah tersimpan ke dalam tabel buku.



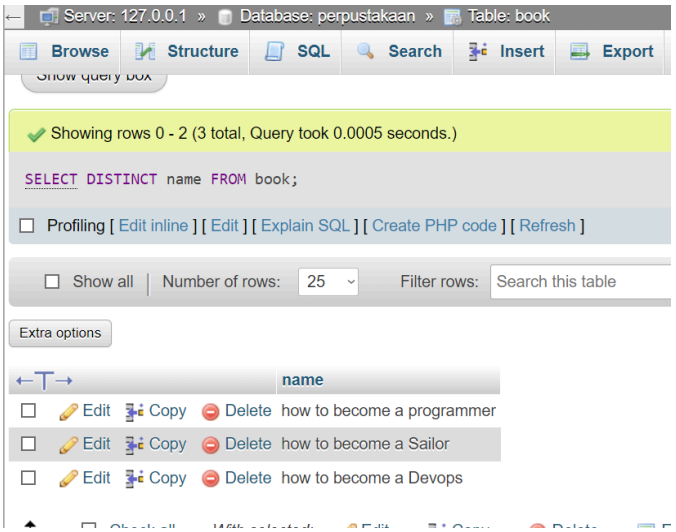
Response Data Telah Disimpan.

4. Lalu kita coba tampilkan semua data yang ada di dalam tabel book namun kita tambahkan *DISTINCT* pada kolom *name* seperti ini.



Query *DISTINCT*.

5. Tekan "*Go*", data yang sama akan tidak ditampilkan seperti ini.



Tampilan Data Setelah Diberi *DISTINCT*.

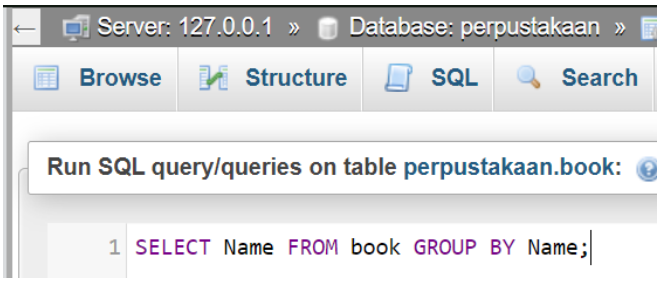
6. GROUP BY.

Ketika kita membuat query *SELECT* kita perlu dikelompokkan baris berdasarkan nilai yang ditentukan dari satu atau banyak kolom, karena di beberapa situasi terdapat data dengan nilai yang sama, dengan dikelompokkan berdasarkan nilai dari kolom tertentu kita dapat melakukan aggregate atau melakukan operasi perhitungan seperti *SUM, COUNT, MIN, MAX, AVG*. dengan begitu kita tidak perlu melihat banyak baris data dengan nilai yang sama. Contoh syntax atau penulisan query *GROUP BY* seperti ini:

```
SELECT (nama_kolom, nama_kolom)
FROM (nama_tabel)
GROUP BY (nama_kolom, nama_kolom);
```

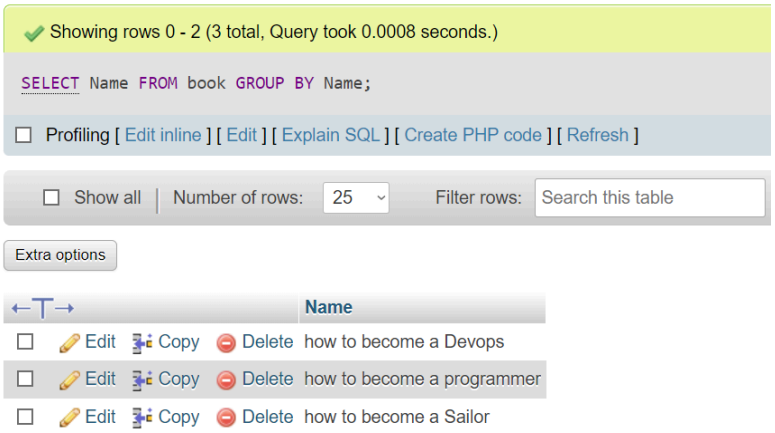
Kita langsung praktekan saja, ikuti langkah - langkah dibawah ini :

1. Pertama kita kembali ke “SQL”.
2. Lalu kita coba tampilkan semua data yang ada di dalam tabel book namun kolom yang akan ditampilkan hanya namanya saja, lalu kita tambahkan *GROUP BY* dan diikuti dengan kolom name, seperti ini.



Query *Group By*.

3. Lalu tekan “*Go*” maka akan menampilkan data seperti ini.

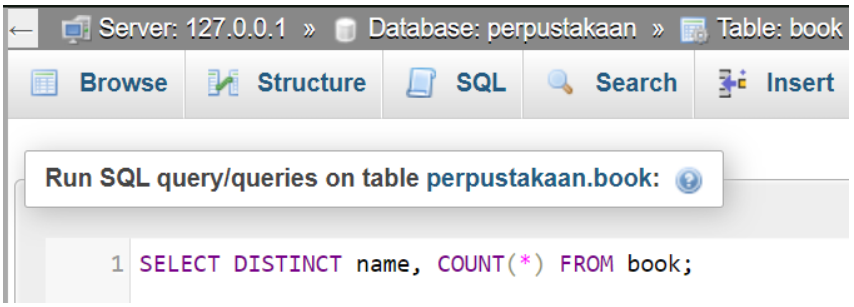


Tampilan Data Setelah Diberi *Group By*.

sebelumnya kita sudah menambahkan data didalam tabel book namun dengan nilai yang sama, Jika kita perhatikan hasil dari query di atas ketika kita beri *GROUP BY* sekilas hasilnya sama seperti kita menggunakan *DISTINCT* tapi sebenarnya itu berbeda, kita coba dengan menghitung berapa jumlah setiap data yang sama dari tabel ini menggunakan jika kita menggunakan *DISTINCT* dan *GROUP BY,* untuk

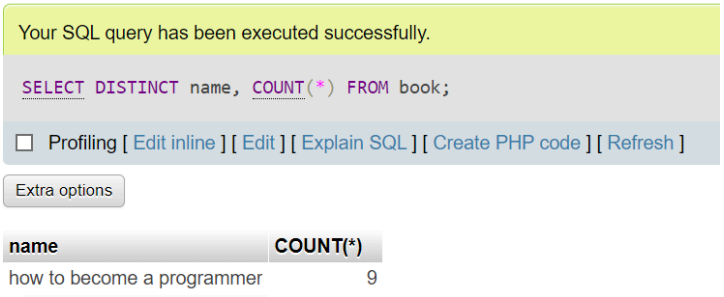
menghitung berapa jumlah di dalam SQL kita dapat menggunakan perintah *COUNT*, ikuti langkah - langkah dibawah ini:

1. Kita kembali lagi ke “SQL”.
2. Lalu kita coba pakai DISTINCT dan diikuti dengan kolom name, lalu setelahnya kita tambahkan COUNT(*) untuk menghitung berapa banyak data yang ada di dalam tabel, query nya menjadi seperti ini.



Query Percobaan Menggunakan *Distinct*.

3. Tekan “Go” dan akan menampilkan data seperti ini.



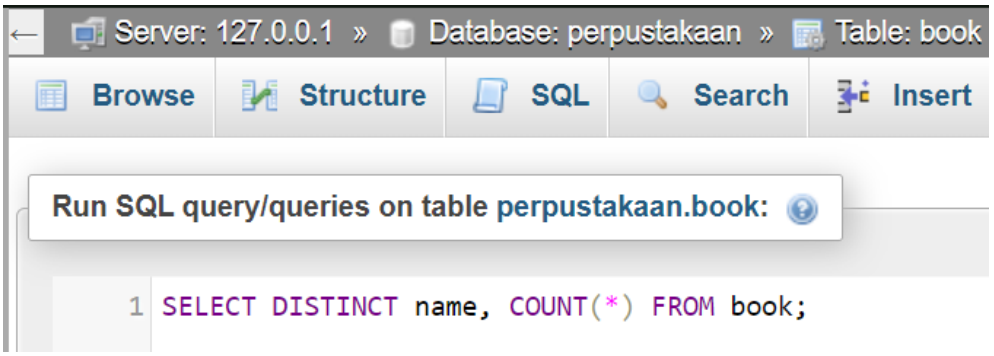
Tampilan Data Menggunakan *DISTINCT*.

Jika kita perhatikan, ketika kita tambahkan *COUNT* setelah *DISTINCT* yang muncul hanya satu baris data saja dan yang dijumlah malah data secara keseluruhan.

Sekarang kita coba menggunakan *GROUP BY*, ikuti langkah - langkah dibawah ini :

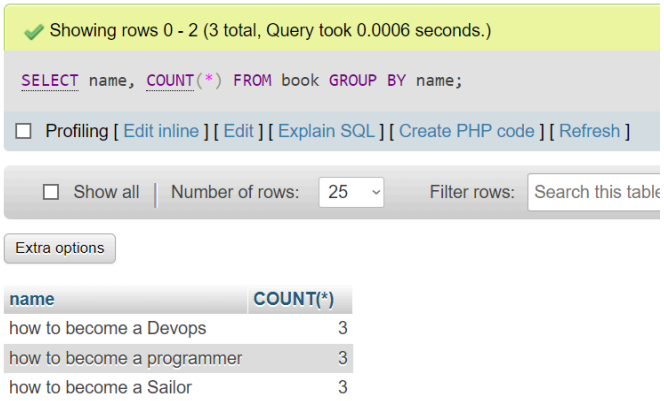
1. Kita kembali lagi ke “SQL”.
2. Lalu kita coba pakai GROUP BY dan diikuti dengan kolom name, lalu kolom yang akan ditampilkan hanya kolom name saja setelahnya kita tambahkan

COUNT(*) untuk menghitung berapa banyak data yang ada di dalam tabel, query nya menjadi seperti ini.



Query Percobaan Menggunakan *Group By*.

3. Tekan "Go" dan akan menampilkan data seperti ini.



Tampilan Data Ketika Menggunakan Group By.

Jika kita menggunakan *Group By* yang dijumlah hanya data yang sama saja sedangkan ketika kita menggunakan *DISTINCT* yang dijumlahkan seluruh data yang ada di dalam tabel itu.

7. HAVING.

Kita sudah mencoba membuat query menggunakan *GROUP BY*, biasanya kita melakukan filtering menggunakan *WHERE*, namun ada kekurangan jika kita menggunakan *WHERE*, kondisi yang ada di dalam perintah WHERE hanya dari nilai pada kolom dan tidak dapat menggunakan data yang sudah di aggregate. Perintah *HAVING* digunakan untuk filtering data yang sudah diakumulasi atau sudah di aggregate menggunakan *SUM, COUNT, MIN, MAX, AVG*. Contoh penggunaan perintah *HAVING* seperti ini :

```
SELECT (nama_kolom, nama_kolom)
FROM (nama_tabel)
GROUP BY (nama_kolom, nama_kolom)
HAVING (aggregate) (operator) (nilai);
```

Untuk menggunakan *HAVING* kita perlu mengelompokan terlebih dahulu sehingga kita perlu menggunakan perintah *GROUP BY* terlebih dahulu sebelum menggunakan perintah *HAVING*, kita juga dapat memberikan perintah *WHERE* jika perlu melakukan filter berdasarkan nilai yang ada di kolom, untuk memberikan perintah where seperti ini.

```
SELECT (nama_kolom, nama_kolom)
FROM (nama_tabel)
WHERE (kondisi)
GROUP BY (nama_kolom, nama_kolom)
HAVING (kondisi);
```

Kita langsung praktekkan saja, ikuti langkah - langkah dibawah ini:

1. Pertama kita klik “SQL”
2. Sebelum kita coba praktekan, kita buat lagi dengan nama yang sama di dalam tabel book.

```
INSERT INTO
    book (Name)
        VALUES
            ("learn how to learn"),
            ("learn how to learn"),
            ("Sikancil");
```

Query *Insert* Data.

3. lalu tekan "*Go*" pastikan data sudah ditambahkan.

←T→				ID	Name
☐	Edit	Copy	Delete	1	how to become a programmer
☐	Edit	Copy	Delete	2	how to become a programmer
☐	Edit	Copy	Delete	3	how to become a programmer
☐	Edit	Copy	Delete	4	how to become a Sailor
☐	Edit	Copy	Delete	5	how to become a Sailor
☐	Edit	Copy	Delete	6	how to become a Sailor
☐	Edit	Copy	Delete	7	how to become a Devops
☐	Edit	Copy	Delete	8	how to become a Devops
☐	Edit	Copy	Delete	9	how to become a Devops
☐	Edit	Copy	Delete	10	learn how to learn
☐	Edit	Copy	Delete	11	learn how to learn
☐	Edit	Copy	Delete	12	Sikancil

Tampilan Tabel *Book* Setelah Ditambahkan Data Baru.

4. Lalu kita coba pakai *GROUP BY* dan diikuti dengan kolom name, lalu kolom yang akan ditampilkan hanya kolom name saja setelahnya kita tambahkan *COUNT*(*) untuk menghitung berapa banyak data yang ada di dalam tabel, query nya menjadi seperti ini. lalu kita ingin data yang ditampilkan itu data yang sama lebih dari 2 jika kurang dari itu maka tidak akan ditampilkan,

Run SQL query/queries on table perpustakaan.book:

```
1 SELECT Name, COUNT(*)
2     FROM book
3         GROUP BY Name
4             HAVING COUNT(*) > 2;
```

Query *Having*.

5. lalu tekan "*Go*" maka data yang akan tampil seperti ini.

Showing rows 0 - 2 (3 total, Query took 0.0005 seconds.)

```
SELECT Name, COUNT(*) FROM book GROUP BY Name HAVING COUNT(*) > 2;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

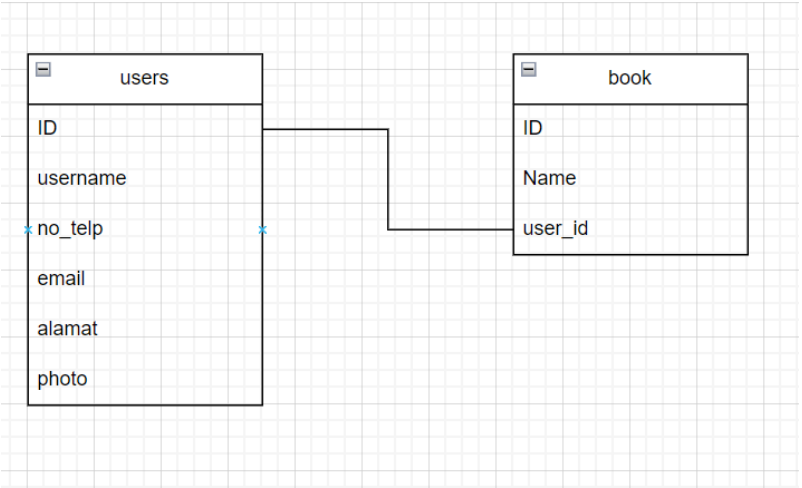
Extra options

Name	COUNT(*)
how to become a Devops	3
how to become a programmer	3
how to become a Sailor	3

Tampilan Data Setelah Diberi *Having*.

8. JOIN.

Jika kita melihat dari schema database yang pernah kita buat ada beberapa tabel yang saling berhubungan dengan tabel lain, misalnya seperti yang ada di dalam struktur ERD sederhana dibawah ini.



Gambar Tabel Yang Saling Berhubungan.

Ketika terdapat tabel yang saling berhubungan dengan tabel lain kita dapat mengambil data dari banyak tabel sekaligus dalam satu kali membuat query, jadi di dalam Query *SELECT* kita dapat mengambil data yang berasal dari tabel yang lain menggunakan perintah *JOIN*,

Ada beberapa jenis join yang umum digunakan dalam SQL, antara lain *INNER JOIN*, *LEFT JOIN*, dan *RIGHT JOIN*.

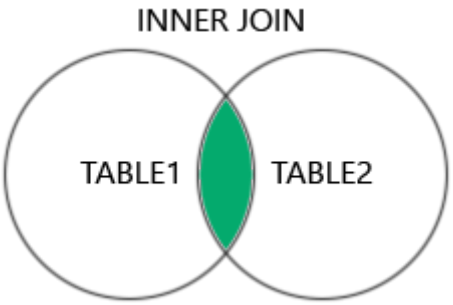
Untuk membuat query *JOIN*, kita perlu menentukan tabel yang akan ditampilkan dan juga kolom yang akan digunakan untuk menghubungkannya dengan tabel lainnya. Gunakan titik untuk menghindari keambiguan dalam penamaan kolom, menunjukkan asal kolom dari tabel mana. Contoh penggunaan *JOIN* adalah sebagai berikut:

```
SELECT (nama_tabel.nama_kolom, nama_tabel.nama_kolom)
FROM (nama_tabel1)
(INNER JOIN / LEFT JOIN / RIGHT JOIN)
     (nama_tabel2) ON (nama_tabel2.nama_kolom) =
(nama_tabel1.nama_kolom);
```

Perintah JOIN ada banyak jenis yang memiliki karakteristiknya masing - masing dan dapat digunakan sesuai dengan kebutuhan kita :

1. **INNER JOIN**

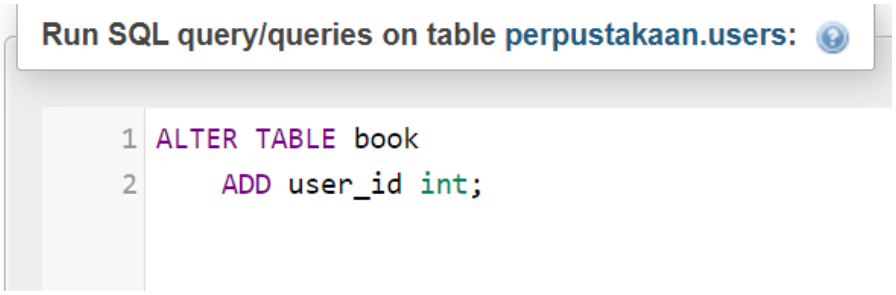
INNER JOIN digunakan untuk mengambil data dari tabel lain yang memenuhi kondisi tertentu. Kondisi ini berarti nilai yang sesuai antara kedua tabel. Sebagai contoh, jika pada tabel "users" terdapat ID dengan nilai 1, dan pada tabel "books" terdapat data buku dengan user_id 1, maka data yang sesuai atau tidak kosong akan ditampilkan.



Gambar Analogi *INNER JOIN*.

Kita langsung praktekkan saja, Ikuti langkah - langkah berikut ini :

1. Pertama kita klik “SQL”
2. Kita tambahkan kolom baru di tabel book user_id, query nya seperti ini.



Query Membuat Kolom Baru

3. Tekan “Go” dan pastikan kolom baru di tabel book sudah ditambahkan.

Server: 127.0.0.1 » Database: perpustakaan » Table: book

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations

Table structure | Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	ID	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	Name	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	user_id	int(11)			Yes	NULL			Change Drop More

Tabel book Yang Telah Ditambahkan Kolom Baru.

4. Lalu kita buat data baru kedalam tabel book.

Run SQL query/queries on table perpustakaan.book:

```
INSERT INTO
    book (Name, user_id)
        VALUES
            ("ini buku baru", 4),
            ("ini buku lama", 1);
```

Query *Insert* Data.

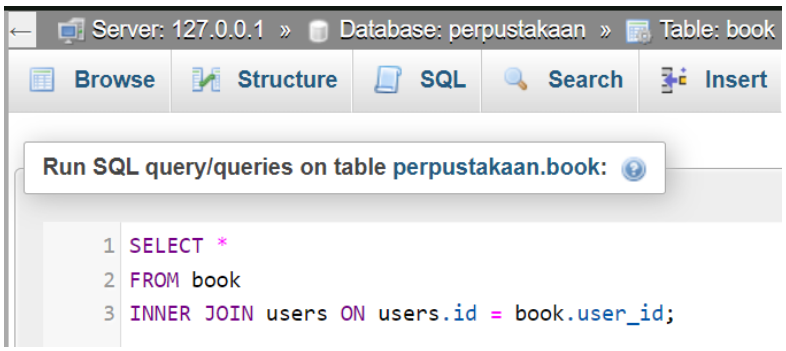
5. Tekan “Go” lalu pastikan data tersimpan didalam tabel.

Browse | Structure | SQL | Search | Insert | Export

	ID	Name	user_id
Edit Copy Delete	3	how to become a programmer	NULL
Edit Copy Delete	4	how to become a Sailor	NULL
Edit Copy Delete	5	how to become a Sailor	NULL
Edit Copy Delete	6	how to become a Sailor	NULL
Edit Copy Delete	7	how to become a Devops	NULL
Edit Copy Delete	8	how to become a Devops	NULL
Edit Copy Delete	9	how to become a Devops	NULL
Edit Copy Delete	10	learn how to learn	NULL
Edit Copy Delete	11	learn how to learn	NULL
Edit Copy Delete	12	Sikancil	NULL
Edit Copy Delete	13	ini buku baru	4
Edit Copy Delete	14	ini buku lama	1

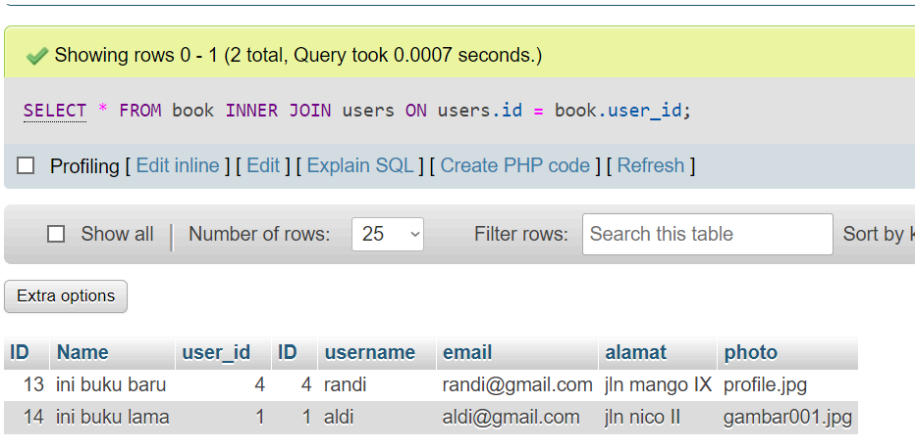
Tampilan Tabel Book Setelah Ditambahkan Data Baru.

6. Lalu kita tampilkan semua data dari tabel *book* dan *users* menggunakan *INNER JOIN*.



Query *INNER JOIN*.

7. Tekan "Go", maka data yang ditampilkan akan seperti ini.

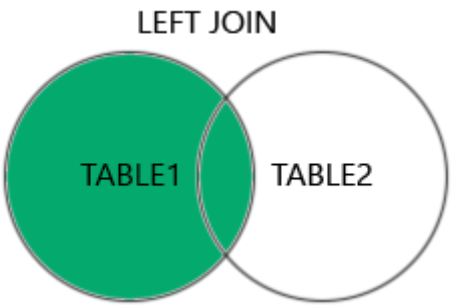


ID	Name	user_id	ID	username	email	alamat	photo
13	ini buku baru	4	4	randi	randi@gmail.com	jln mango IX	profile.jpg
14	ini buku lama	1	1	aldi	aldi@gmail.com	jln nico II	gambar001.jpg

Hasil Dari Query *INNER JOIN*.

2. LEFT JOIN

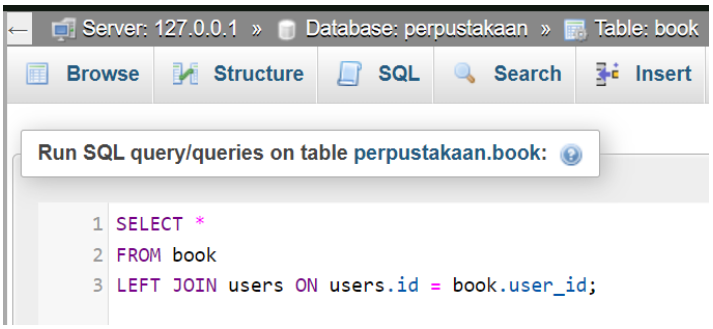
LEFT JOIN digunakan untuk mengembalikan semua baris tabel di sisi kiri gabungan dan mencocokkan baris tabel di sisi kanan gabungan. Untuk baris yang tidak ada baris yang cocok di sisi kanannya, data akan tetap ditampilkan namun nilainya akan nol.



Gambar Analogi *LEFT JOIN*.

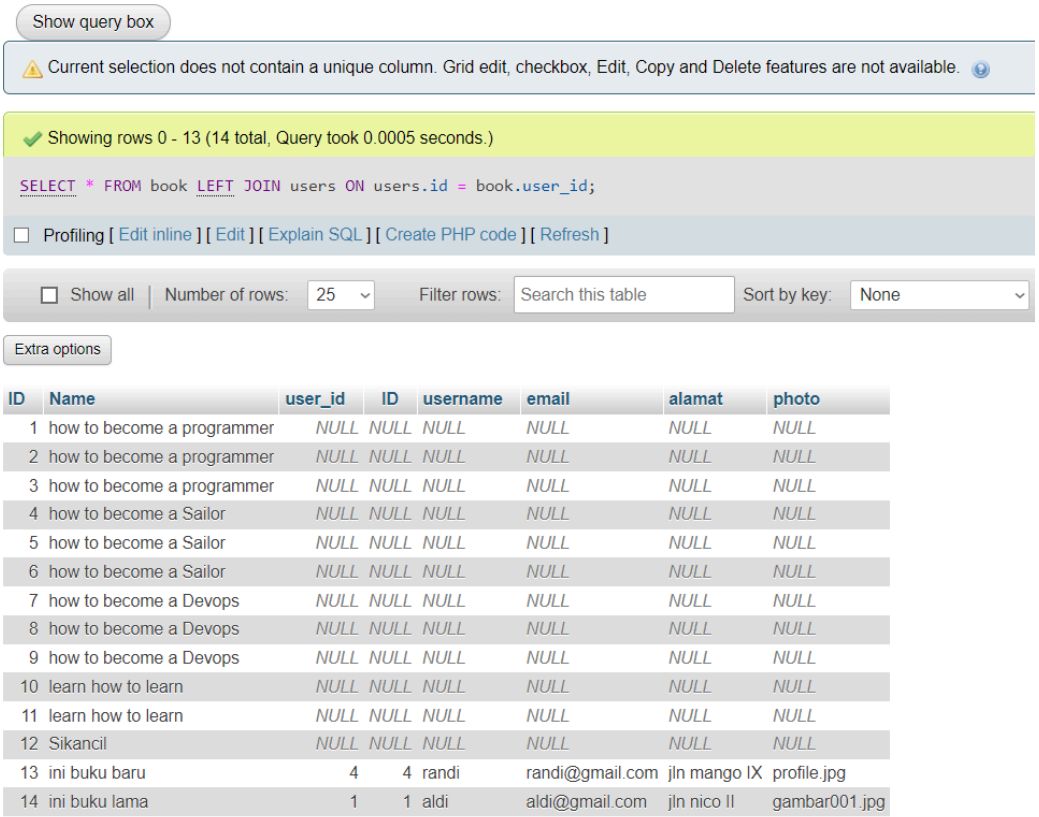
Kita langsung praktekkan saja, Ikuti langkah - langkah berikut ini :

1. Pertama kita klik “SQL”
2. Lalu kita tampilkan semua data dari tabel *book* dan *users* menggunakan *LEFT JOIN*.



Query *Left Join*.

3. Tekan “Go”, maka data yang ditampilkan akan seperti ini.

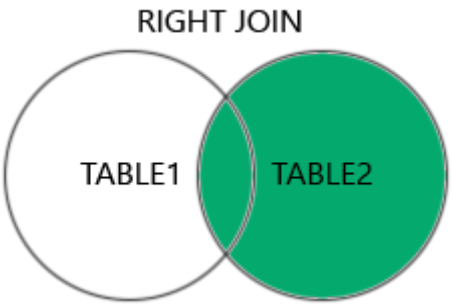


ID	Name	user_id	ID	username	email	alamat	photo
1	how to become a programmer	NULL	NULL	NULL	NULL	NULL	NULL
2	how to become a programmer	NULL	NULL	NULL	NULL	NULL	NULL
3	how to become a programmer	NULL	NULL	NULL	NULL	NULL	NULL
4	how to become a Sailor	NULL	NULL	NULL	NULL	NULL	NULL
5	how to become a Sailor	NULL	NULL	NULL	NULL	NULL	NULL
6	how to become a Sailor	NULL	NULL	NULL	NULL	NULL	NULL
7	how to become a Devops	NULL	NULL	NULL	NULL	NULL	NULL
8	how to become a Devops	NULL	NULL	NULL	NULL	NULL	NULL
9	how to become a Devops	NULL	NULL	NULL	NULL	NULL	NULL
10	learn how to learn	NULL	NULL	NULL	NULL	NULL	NULL
11	learn how to learn	NULL	NULL	NULL	NULL	NULL	NULL
12	Sikancil	NULL	NULL	NULL	NULL	NULL	NULL
13	ini buku baru	4	4	randi	randi@gmail.com	jln mango IX	profile.jpg
14	ini buku lama	1	1	aldi	aldi@gmail.com	jln nico II	gambar001.jpg

Tampilan Hasil Dari Query *Left Join*.

3. RIGHT JOIN

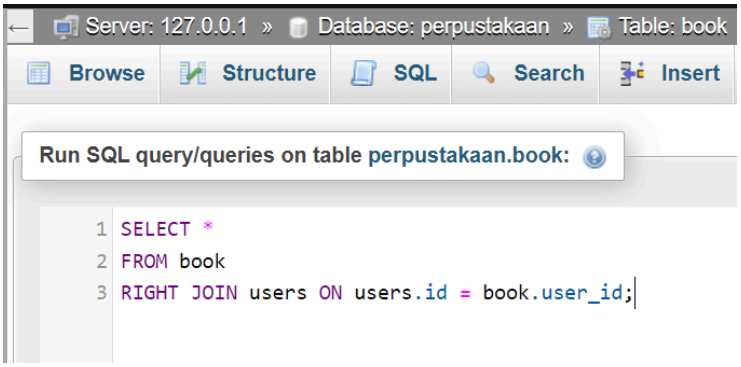
RIGHT JOIN sama seperti *LEFT JOIN*, digunakan untuk menampilkan semua baris tabel di sisi kanan gabungan dan baris yang cocok untuk tabel di sisi kiri gabungan. Untuk baris yang tidak ada baris yang cocok di sisi kiri, kumpulan hasil akan berisi nol.



Gambar Analogi *RIGHT JOIN*.

Kita langsung praktekkan saja, Ikuti langkah - langkah dibawah ini :

1. Pertama kita klik “SQL”
2. Lalu kita tampilkan semua data dari tabel *book* dan *users* menggunakan *RIGHT JOIN*.



Query *RIGHT JOIN*.

3. Tekan “Go”, maka data yang ditampilkan akan seperti ini.

Showing rows 0 - 10 (11 total, Query took 0.0054 seconds.)

```
SELECT * FROM book RIGHT JOIN users ON users.id = book.user_id;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

ID	Name	user_id	ID	username	email	alamat	photo
13	ini buku baru	4	4	randi	randi@gmail.com	jln mango IX	profile.jpg
14	ini buku lama	1	1	aldi	aldi@gmail.com	jln nico II	gambar001.jpg
NULL	NULL	NULL	2	ahmad	ahmad@gmail.com	jln nestar VI	gambar.jpg
NULL	NULL	NULL	3	rico	rico@gmail.com	jln cookie I	profile.jpg
NULL	NULL	NULL	5	aldo	aldo@gmail.com	jln markisa VIII	profile.jpg
NULL	NULL	NULL	6	kevin	kevin@gmail.com	jln apel V	profile.jpg
NULL	NULL	NULL	7	fauzan	fauzan@gmail.com	jln Bayam XIV	profile.jpg
NULL	NULL	NULL	8	aulia	aulia@gmail.com	jln jeruk XVIII	profile.jpg
NULL	NULL	NULL	9	riko	riko@gmail.com	jln jeruk XVIII	profile.jpg
NULL	NULL	NULL	10	aldo	aldo@gmail.com	jln markisa VIII	profile.jpg
NULL	NULL	NULL	11	kevin	kevin@gmail.com	jln apel V	profile.jpg

Tampilan Hasil Query *RIGHT JOIN*.

9. SUBQUERY.

Terkadang di beberapa kondisi kita perlu data dengan kondisi yang memerlukan operasi pada dua tabel atau lebih, di dalam query SQL terdapat sebuah konsep yang bernama Subquery, yakni query yang terdapat di dalam query utama dalam SQL.

Cara kerja subquery yaitu system akan menjalankan Subquery terlebih dahulu sebelum query utama dijalankan, sehingga membuat subquery memerlukan resource yang besar, sehingga proses menjalankan query akan lebih lambat. Untuk membuat subquery yaitu dengan menambahkan kurung buka kurung tutup yang didalamnya berisi query biasa, subquery dapat digunakan sebagai nilai pembanding dalam kondisi *Where* atau ditampilkan sebagai kolom, penggunaan Subquery pada query utama seperti ini :

Contoh subquery yang didefinisikan pada kolom yang akan ditampilkan.

```
SELECT (subquery) FROM (nama_tabel);
```

Contoh subquery yang didefinisikan pada kondisi *Where*.

```
SELECT (nama_kolom)
FROM (nama_tabel)
```

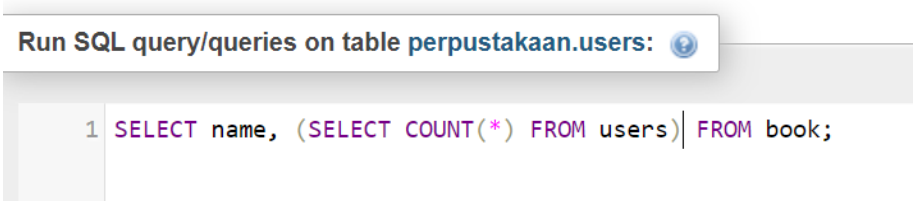
```
WHERE (nama_kolom) (operator) (subquery);
```

Contoh query di dalam subquery.

```
SELECT (nama_kolom) FROM (nama_tabel);
```

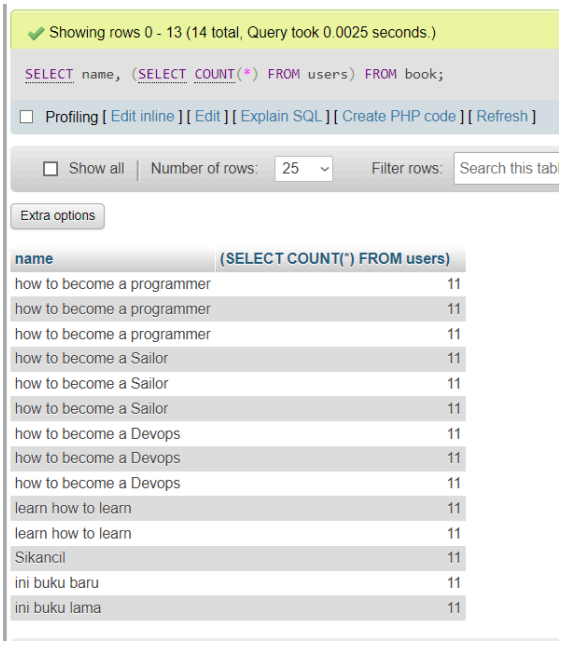
Kita langsung praktekkan saja, ikuti langkah - langkah dibawah ini :

1. Pertama kita tampilkan data di dalam book namun kita tampilkan berapa jumlah user yang ada didalam tabel users, seperti ini.



Subquery.

2. Lalu jalankan maka akan menampilkan data seperti ini.



name	(SELECT COUNT(*) FROM users)
how to become a programmer	11
how to become a programmer	11
how to become a programmer	11
how to become a Sailor	11
how to become a Sailor	11
how to become a Sailor	11
how to become a Devops	11
how to become a Devops	11
how to become a Devops	11
learn how to learn	11
learn how to learn	11
Sikancil	11
ini buku baru	11
ini buku lama	11

Hasil Dari Subquery Yang Telah Dijalankan.

Kita sudah membuat subquery sederhana yang menampilkan seluruh data yang ada didalam tabel *book* beserta dengan seluruh data yang ada didalam tabel users, sekarang kita akan mencoba membuat subquery sebagai nilai pembanding di dalam kondisi *Where*. ikuti langkah - langkah dibawah ini :

1. Pertama kita tampilkan data di dalam book namun yang akan ditampilkan yaitu id buku yang kurang dari nilai subquery, seperti ini.

Run SQL query/queries on table perpustakaan.users:

```
1 SELECT name
2     FROM book
3         WHERE id < (SELECT id FROM users WHERE username = "kevin");
```

Query Subquery.

2. Lalu jalankan maka akan menampilkan data seperti ini.

Showing rows 0 - 4 (5 total, Query took 0.0005 seconds.)

```
SELECT name FROM book WHERE id < (SELECT id FROM users WHERE username = "kevin");
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key:

Extra options

←T→				name
Edit	Copy	Delete		how to become a programmer
Edit	Copy	Delete		how to become a programmer
Edit	Copy	Delete		how to become a programmer
Edit	Copy	Delete		how to become a Sailor
Edit	Copy	Delete		how to become a Sailor

Hasil Dari Subquery Yang Telah Dijalankan.

Jika kita lihat di dalam subquery berisikan query yang dimana usernamenya kevin lalu id nya dipakai sebagai pembanding dengan kolom id yang di tabel *book*, jika id buku kurang dari nilai subquery yang mana berisikan id dari user bernama kevin maka akan ditampilkan.

C. UPDATE.

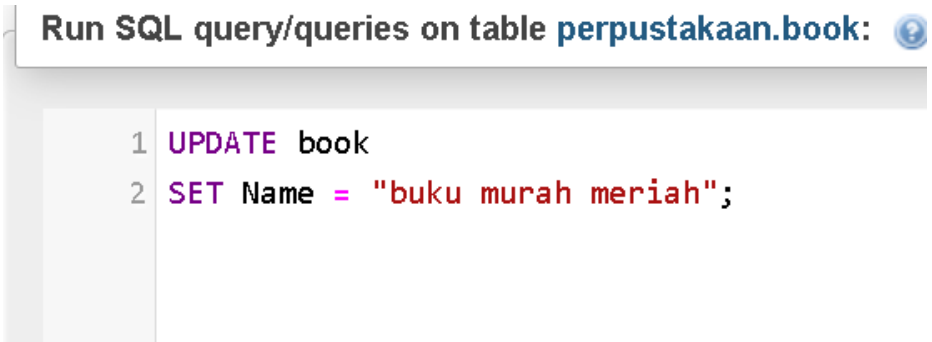
Setelah kita menambahkan data didalam tabel, kita dapat mendapatkan merubah data di dalamnya menggunakan perintah *Update*. perintah *Update* digunakan untuk merubah suatu data tertentu di dalam tabel. Contoh penggunaan perintah *Update* seperti ini:

```
UPDATE (nama_tabel)
SET (nama_kolom = value1, nama_kolom = value2);
```

kita juga dapat menambahkan kondisi agar data yang berubah hanya data tertentu saja. Contoh penggunaan perintah *Update* dengan menambahkan kondisi seperti ini:

```
UPDATE (nama_tabel)
SET (nama_kolom = value, nama_kolom = value)
WHERE (kondisi);
```

Kita coba jalankan perintah *Update* seperti dibawah ini :



Query *Update* Tanpa Kondisi.

Setelah kita jalankan query di atas maka seluruh data didalam tabel akan berubah.

←T→			ID	Name	user_id
Edit	Copy	Delete	1	buku murah meriah	NULL
Edit	Copy	Delete	2	buku murah meriah	NULL
Edit	Copy	Delete	3	buku murah meriah	NULL
Edit	Copy	Delete	4	buku murah meriah	NULL
Edit	Copy	Delete	5	buku murah meriah	NULL
Edit	Copy	Delete	6	buku murah meriah	NULL
Edit	Copy	Delete	7	buku murah meriah	NULL
Edit	Copy	Delete	8	buku murah meriah	NULL
Edit	Copy	Delete	9	buku murah meriah	NULL
Edit	Copy	Delete	10	buku murah meriah	NULL
Edit	Copy	Delete	11	buku murah meriah	NULL
Edit	Copy	Delete	12	buku murah meriah	NULL
Edit	Copy	Delete	13	buku murah meriah	4
Edit	Copy	Delete	14	buku murah meriah	1

Tampilan Data Setelah Di *Update*.

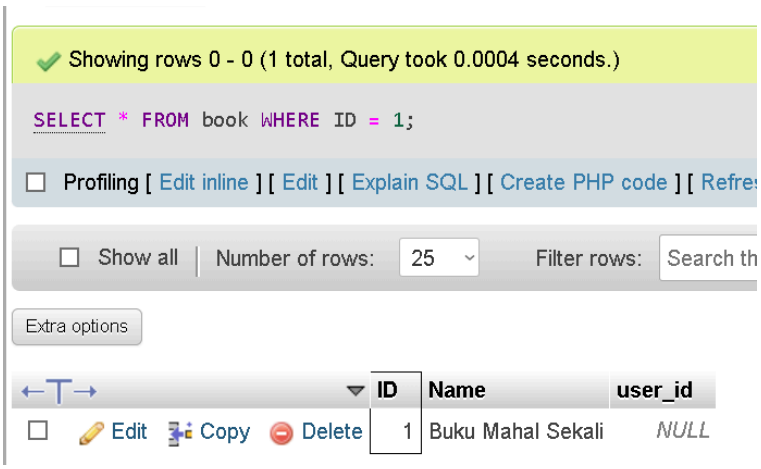
Jika kita ingin mengubah data tertentu saja kita dapat menggunakan *Where,* Misalnya kita akan ubah data buku menjadi "buku mahal sekali" yang idnya 1, maka perintahnya sebagai berikut:

Run SQL query/queries on table perpustakaan.book:

```
1 UPDATE book
2 SET Name = "Buku Mahal Sekali"
3 WHERE ID = 1;
```

Query *Update* Dengan Perintah *Where*.

Setelah itu tekan "*Go*", apabila berhasil maka data dengan idnya 1 maka akan berubah nilai.



Tampilan Data Setelah Di *Update*.

D. DELETE

Ketika data kita sudah mulai banyak atau ukuran database sudah membesar, maka kita perlu menghapus beberapa data di dalam tabel yang tidak terpakai. Untuk menghapus data di dalam tabel kita dapat menggunakan perintah *Delete*, Fungsi *Delete* ini dapat digunakan untuk menghapus satu atau lebih data. Contoh penggunaan perintah *Delete* seperti dibawah ini :

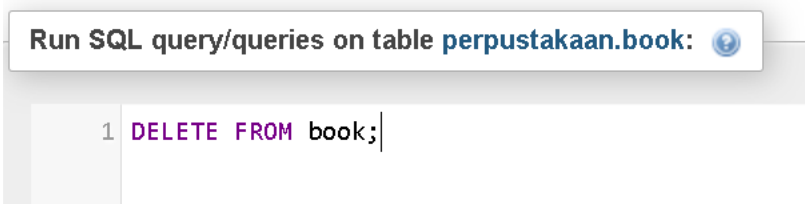
```
DELETE FROM (nama_tabel);
```

Contoh diatas jika dijalankan akan menyebabkan menghapus semua data didalam tabel yg didefinisikan. Jika kita ingin menghapus beberapa data saja, kita dapat menambahkan *Where* lalu berikan kondisinya seperti apa setelah nama tabel, contohnya seperti ini :

```
DELETE FROM (nama_tabel) WHERE (Kondisi);
```

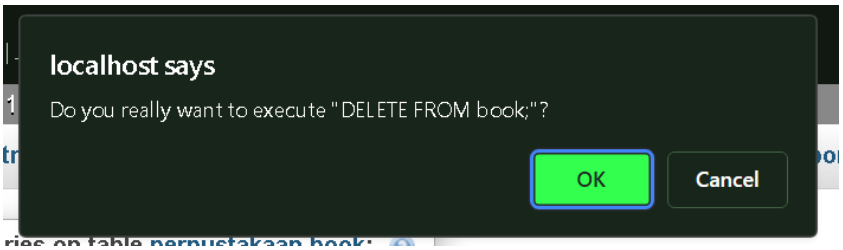
Kita langsung praktekkan saja, Ikuti langkah langkah dibawah ini :

1. Pertama kita coba hapus semua data yang ada didalam tabel *book*, perintah Querynya seperti ini.



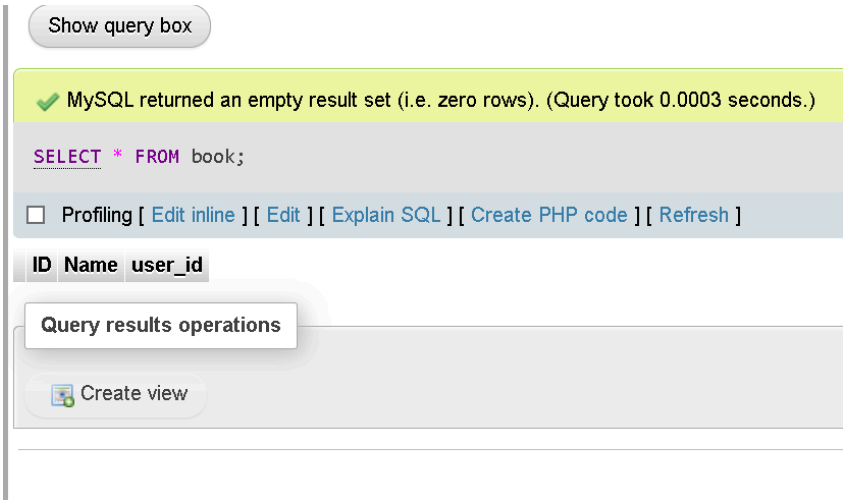
Query Delete Semua Data.

2. Lalu jalankan, jika ada pop up peringatan klik ok.



Konfirmasi Proses Delete.

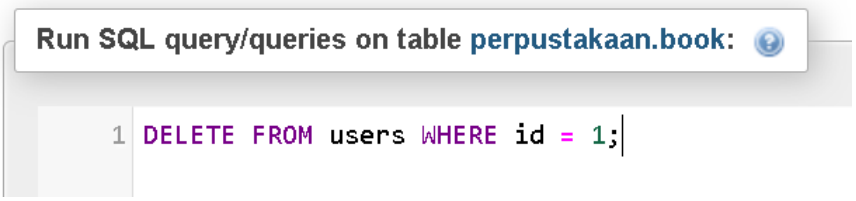
3. Lalu pastikan semua data sudah dihapus.



Tampilan Data Setelah Dihapus Semua.

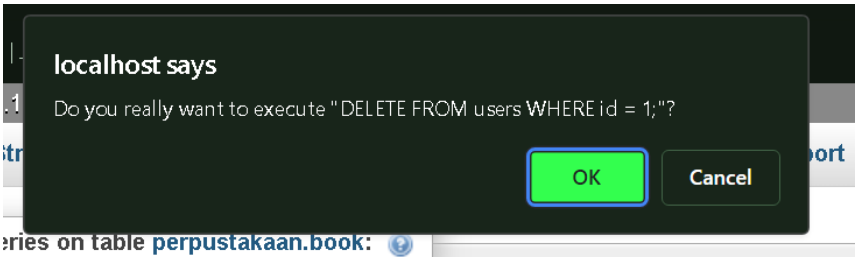
Kita sudah mempraktekkan menghapus semua data yang ada di dalam tabel book. Kita akan mencoba mempraktekkan menghapus beberapa data saja di dalam tabel users.

1. Pertama kita coba beberapa hapus data yang ada di dalam tabel users, saat menggunakan perintah *DELETE* kita tambahkan perintah *WHERE* di dalamnya untuk filter data, seperti ini.



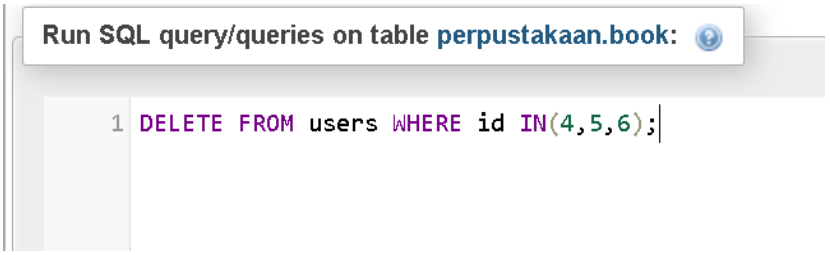
Query *Delete* dengan *Where*.

2. Lalu kita jalankan, jika ada pesan konfirmasi maka klik *ok*.



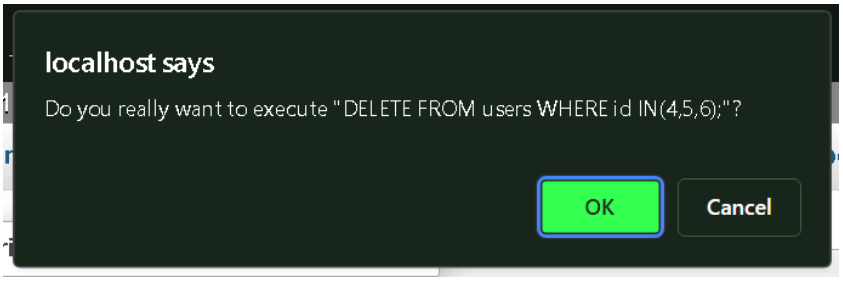
Konfirmasi Proses Delete.

3. Kita juga dapat menghapus banyak data dalam satu perintah, yaitu dengan menggunakan IN, seperti ini.



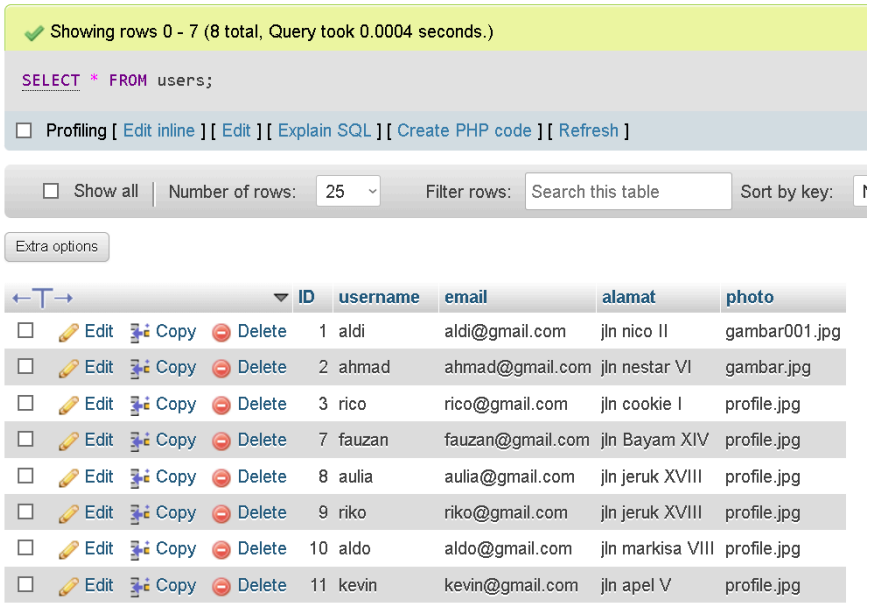
Query Delete dengan *IN* Di dalam Kondisi.

4. Lalu kita jalankan, jika ada pesan konfirmasi maka klik *ok*.



Konfirmasi Proses *Delete*.

5. Lalu kita pastikan data yang sudah kita delete sudah terhapus.



Tampilan Data Didalam Tabel *Users*.

D. Referensi.

https://kelasprogrammer.com/penggunaan-select-distinct-di-mysql/

https://revou.co/panduan-teknis/sql-having

https://revou.co/panduan-teknis/sql-group-by

https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-joins/

https://www.w3schools.com/sql/sql_join.asp