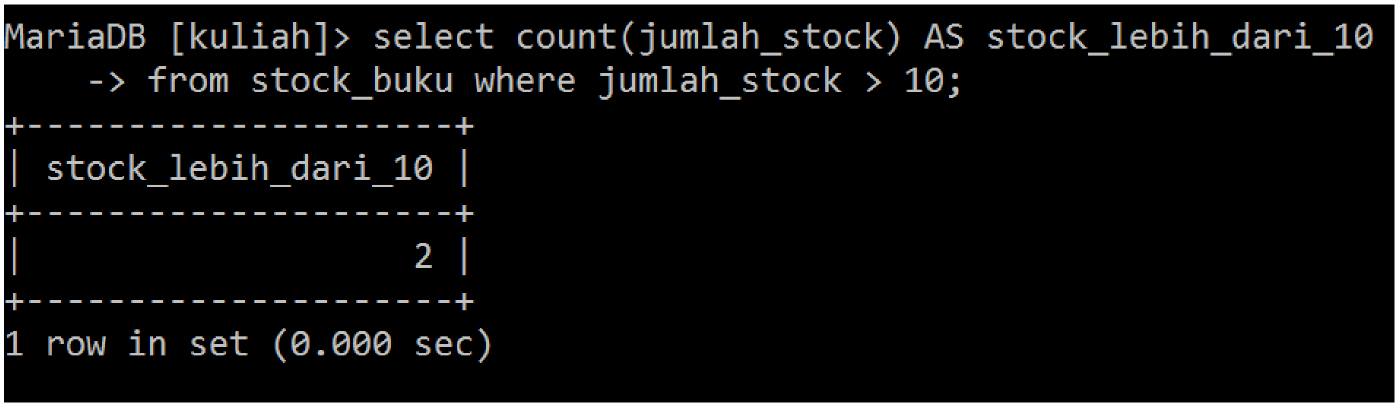
AGGREGATE FUNCTION& GROUP

Buat database, tabel, dan isi tabel :

```
CREATE DATABASE kuliah;


USE kuliah;


CREATE TABLE stock_buku (
    id_buku INT PRIMARY KEY,
    judul_buku VARCHAR(100),
    jumlah_stock INT
);


INSERT INTO stock_buku (id_buku, judul_buku, jumlah_stock) VALUES
(1, 'Principles and Practices of Interconnection Networks', 10),
(2, 'Building DMZs for Enterprise Networks', 20),
(3, 'Virtual Private Networks', 11),
(4, 'Computer Networking: a Top Down Approach', 4),
(5, 'Computer Networking Illuminated', NULL),
(6, 'Network Routing Algorithms, Protocols and Architectures', NULL);
```

Fungsi COUNT()

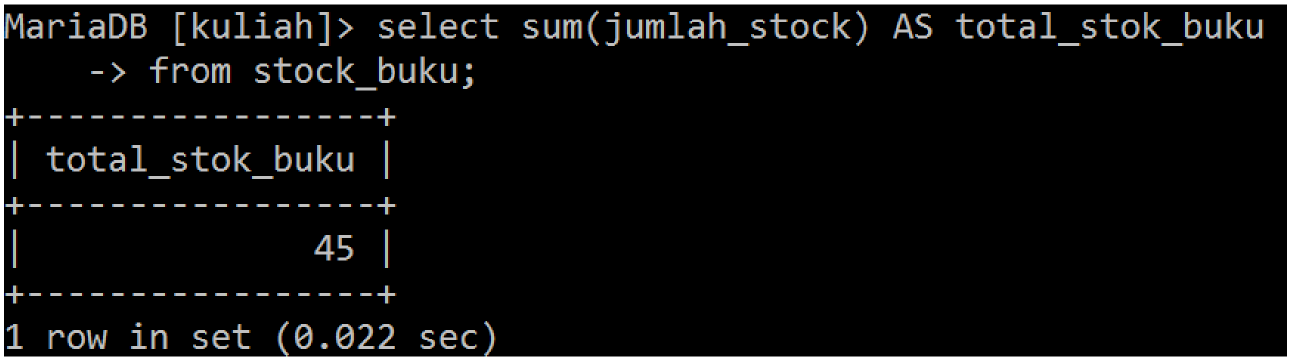
- Fungsi COUNT() digunakan untuk menghitung jumlah baris (data) dalam tabel yang memenuhi kondisi tertentu. Biasanya dipakai untuk mengetahui berapa banyak data yang sesuai dengan suatu kriteria, seperti jumlah buku yang stoknya lebih dari 10, jumlah mahasiswa yang sudah lulus, dan sebagainya.
- Syntax: SELECT COUNT(nama_kolom) FROM nama_tabel;
- Contoh: SELECT COUNT(jumlah_stock) AS stock_lebih_dari_10 FROM stock_buku WHERE jumlah_stock > 10;



```
MariaDB [kuliah]> select count(jumlah_stock) AS stock_lebih_dari_10
    -> from stock_buku where jumlah_stock > 10;
+---------------------+
| stock_lebih_dari_10 |
+---------------------+
|                   2 |
+---------------------+
1 row in set (0.000 sec)
```

FUNGSI SUM()

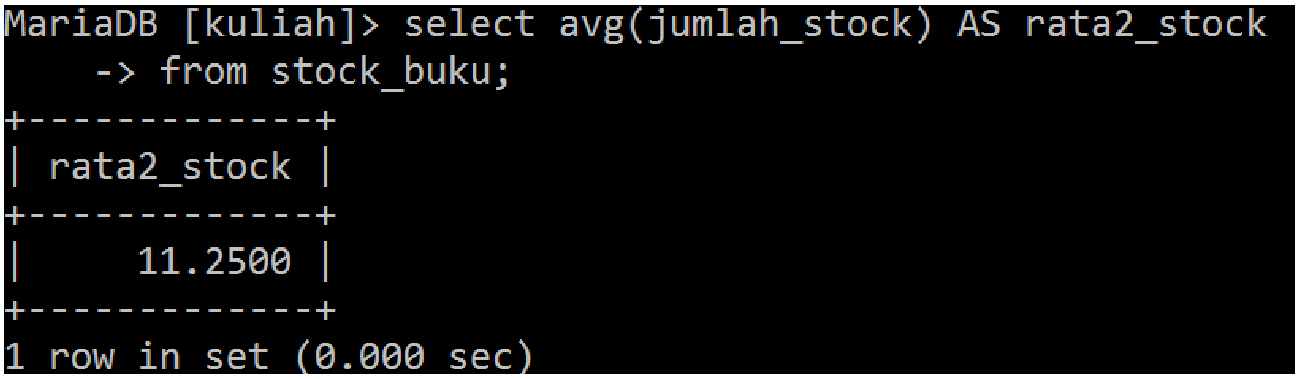
- Fungsi SUM() digunakan untuk menjumlahkan seluruh nilai numeri dalam suatu kolom.
- Syntax: SELECT SUM(nama_kolom) FROM nama_tabel;
- Contoh: SELECT SUM(jumlah_stock) AS total_stok_buku FROM stock_buku;



```
MariaDB [kuliah]> select sum(jumlah_stock) AS total_stok_buku
    -> from stock_buku;
+-----------------+
| total_stok_buku |
+-----------------+
|              45 |
+-----------------+
1 row in set (0.022 sec)
```

Fungsi AVG()

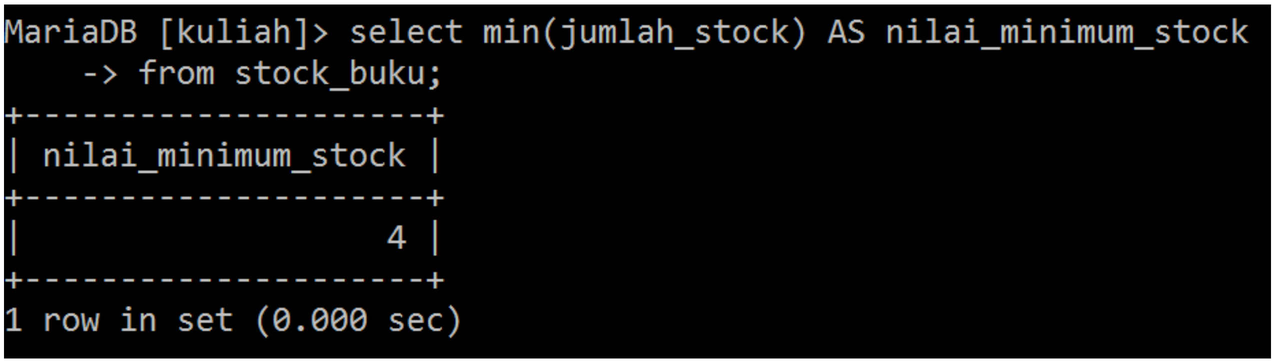
- Fungsi AVG() digunakan untuk digunakan untuk menghitung rata-rata nilai dari suatu kolom.
- Syntax: SELECT AVG(nama_kolom) FROM nama_tabel;
- Contoh: SELECT AVG(jumlah_stock) AS rata2_stock FROM stock_buku;



```
MariaDB [kuliah]> select avg(jumlah_stock) AS rata2_stock
    -> from stock_buku;
+-------------+
| rata2_stock |
+-------------+
|     11.2500 |
+-------------+
1 row in set (0.000 sec)
```

Fungsi MIN()

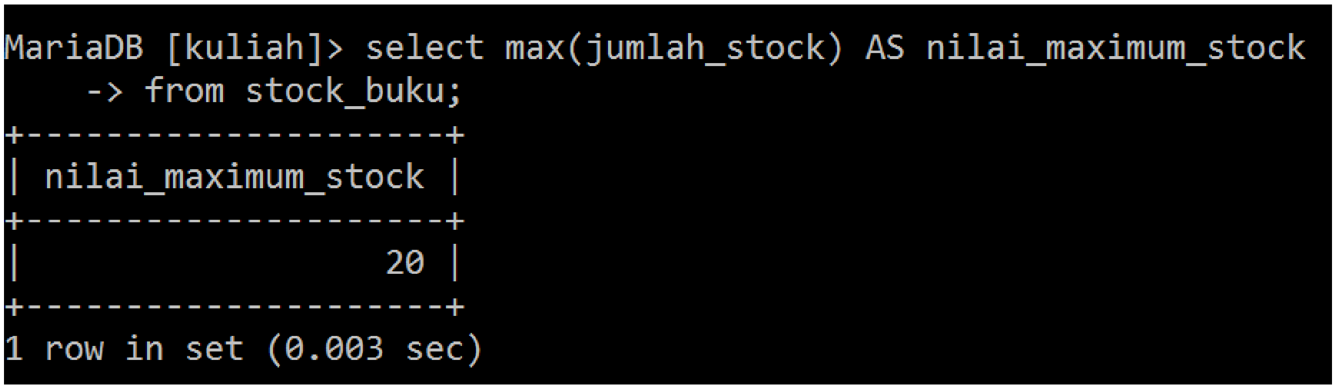
- Fungsi MIN() digunakan untuk digunakan mengembalikan nilai minimum dari suatu kolom.
- Syntax: SELECT MIN(nama_kolom) FROM nama_tabel;
- Contoh: SELECT MIN(jumlah_stock) AS nilai_minimum_stock FROM stock_buku;



```
MariaDB [kuliah]> select min(jumlah_stock) AS nilai_minimum_stock
    -> from stock_buku;
+---------------------+
| nilai_minimum_stock |
+---------------------+
|                   4 |
+---------------------+
1 row in set (0.000 sec)
```

Fungsi MAX()

- Fungsi MAX() digunakan untuk digunakan mengembalikan nilai maximum dari suatu kolom.
- Syntax: SELECT MAX(nama_kolom) FROM nama_tabel;
- Contoh: SELECT MAX(jumlah_stock) AS nilai_maximum_stock FROM stock_buku;



```
MariaDB [kuliah]> select max(jumlah_stock) AS nilai_maximum_stock
    -> from stock_buku;
+---------------------+
| nilai_maximum_stock |
+---------------------+
|                  20 |
+---------------------+
1 row in set (0.003 sec)
```

GROUP BY

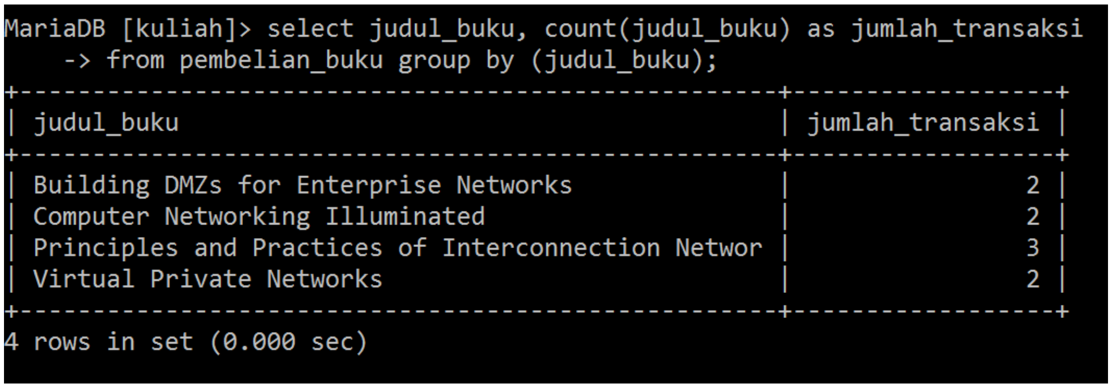
- GROUP BY digunakan mengelompokkan data yang sama dari suatu kolom.
- Pada umumnya GROUP BY digunakan bersamaan dengan aggregate functions seperti COUNT(), SUM(), AVG(), MIN(), dan MAX()
- Syntax: SELECT nama_kolom_1,…,nama_kolom_n FROM nama_tabel GROUP BY nama_kolom_1,…,nama_kolom_n;

Buat Database :

```
CREATE DATABASE kuliah;


USE kuliah;


CREATE TABLE pembelian_buku (
    id_pembelian INT PRIMARY KEY,
    judul_buku VARCHAR(100),
    tanggal_pembelian DATE,
    total_pembelian INT
);


INSERT INTO pembelian_buku (id_pembelian, judul_buku, tanggal_pembelian, total_pembelian)
VALUES
(1, 'Principles and Practices of Interconnection Networ', '2021-07-21', 1),
(2, 'Virtual Private Networks', '2021-07-28', 3),
(3, 'Computer Networking Illuminated', '2021-07-21', 2),
(4, 'Computer Networking Illuminated', '2021-09-30', 5),
(5, 'Virtual Private Networks', '2021-10-12', 4),
(6, 'Building DMZs for Enterprise Networks', '2021-10-15', 10),
(7, 'Building DMZs for Enterprise Networks', '2021-10-19', 10),
(8, 'Principles and Practices of Interconnection Networ', '2021-10-29', 8),
(9, 'Principles and Practices of Interconnection Networ', '2021-11-30', 11);
```

GROUP BY dengan Fungsi COUNT()

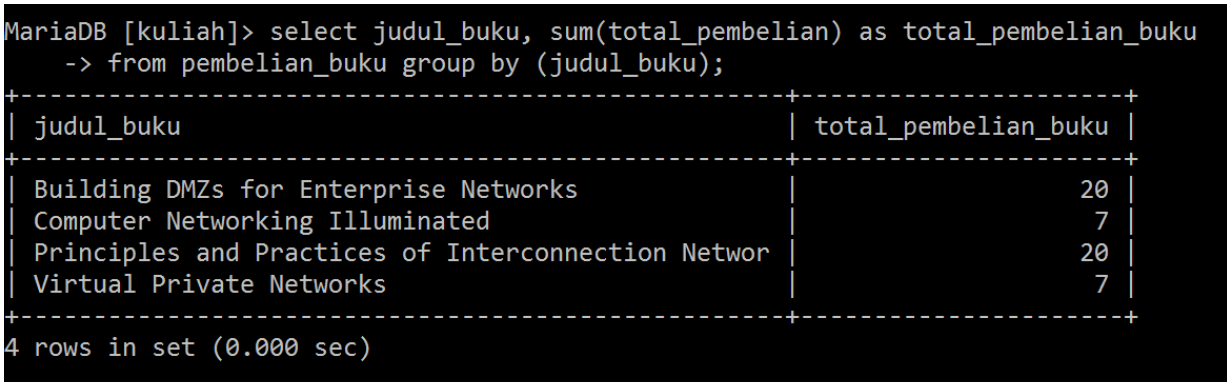
- Contoh: SELECT judul_buku, COUNT(judul_buku) AS jumlah_transaksi FROM pembelian_buku GROUP BY judul_buku;



```
MariaDB [kuliah]> select judul_buku, count(judul_buku) as jumlah_transaksi
    -> from pembelian_buku group by (judul_buku);
+----------------------------------------------------+------------------+
| judul_buku                                         | jumlah_transaksi |
+----------------------------------------------------+------------------+
| Building DMZs for Enterprise Networks              |                2 |
| Computer Networking Illuminated                    |                2 |
| Principles and Practices of Interconnection Networ |                3 |
| Virtual Private Networks                           |                2 |
+----------------------------------------------------+------------------+
4 rows in set (0.000 sec)
```

GROUP BY dengan Fungsi SUM()

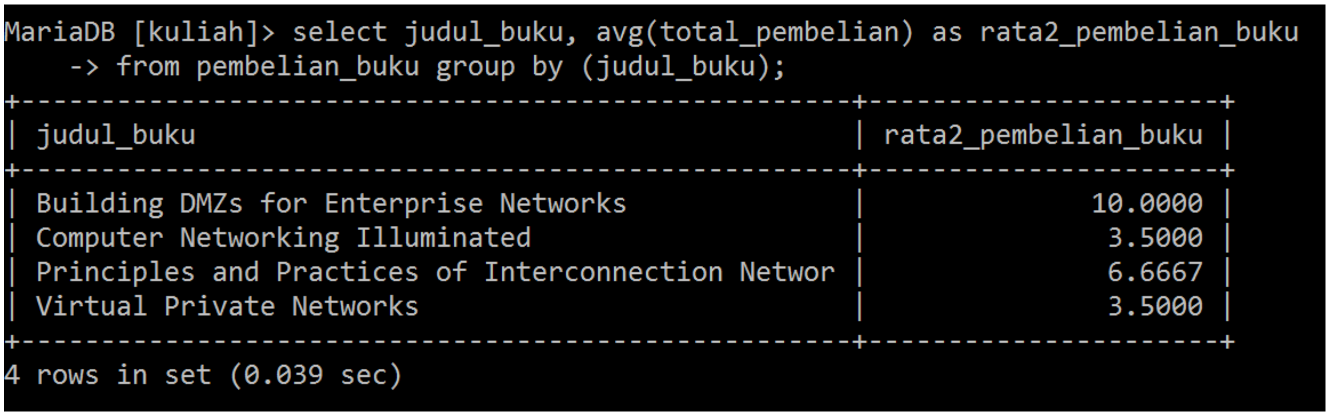
- Contoh: SELECT judul_buku, SUM(total_pembelian) AS total_pembelian_buku FROM pembelian_buku GROUP BY judul_buku;



```
MariaDB [kuliah]> select judul_buku, sum(total_pembelian) as total_pembelian_buku
    -> from pembelian_buku group by (judul_buku);
+----------------------------------------------------+----------------------+
| judul_buku                                         | total_pembelian_buku |
+----------------------------------------------------+----------------------+
| Building DMZs for Enterprise Networks              |                   20 |
| Computer Networking Illuminated                    |                    7 |
| Principles and Practices of Interconnection Networ |                   20 |
| Virtual Private Networks                           |                    7 |
+----------------------------------------------------+----------------------+
4 rows in set (0.000 sec)
```

GROUP BY dengan Fungsi AVG()

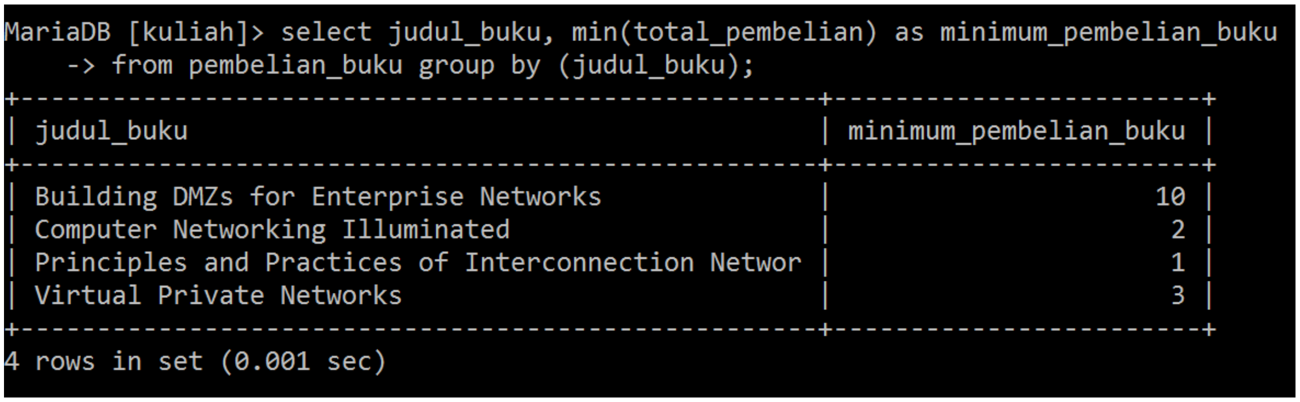
- Contoh: SELECT judul_buku, AVG(total_pembelian) AS rata2_pembelian_buku FROM pembelian_buku GROUP BY judul_buku;



```
MariaDB [kuliah]> select judul_buku, avg(total_pembelian) as rata2_pembelian_buku
    -> from pembelian_buku group by (judul_buku);
+----------------------------------------------------+----------------------+
| judul_buku                                         | rata2_pembelian_buku |
+----------------------------------------------------+----------------------+
| Building DMZs for Enterprise Networks              |              10.0000 |
| Computer Networking Illuminated                    |               3.5000 |
| Principles and Practices of Interconnection Networ |               6.6667 |
| Virtual Private Networks                           |               3.5000 |
+----------------------------------------------------+----------------------+
4 rows in set (0.039 sec)
```

GROUP BY dengan Fungsi MIN()

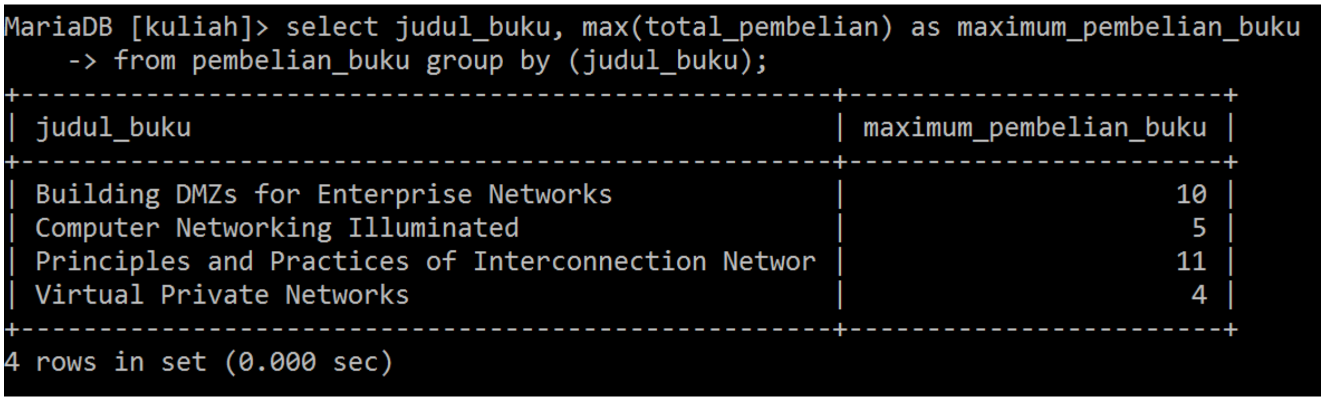
- Contoh: SELECT judul_buku, MIN(total_pembelian) AS minimum_pembelian_buku FROM pembelian_buku GROUP BY judul_buku;



```
MariaDB [kuliah]> select judul_buku, min(total_pembelian) as minimum_pembelian_buku
    -> from pembelian_buku group by (judul_buku);
+----------------------------------------------------+------------------------+
| judul_buku                                         | minimum_pembelian_buku |
+----------------------------------------------------+------------------------+
| Building DMZs for Enterprise Networks              |                     10 |
| Computer Networking Illuminated                    |                      2 |
| Principles and Practices of Interconnection Networ |                      1 |
| Virtual Private Networks                           |                      3 |
+----------------------------------------------------+------------------------+
4 rows in set (0.001 sec)
```

GROUP BY dengan Fungsi MAX()

- Contoh: SELECT judul_buku, MAX(total_pembelian) AS maximum_pembelian_buku FROM pembelian_buku GROUP BY judul_buku;



```
MariaDB [kuliah]> select judul_buku, max(total_pembelian) as maximum_pembelian_buku
    -> from pembelian_buku group by (judul_buku);
+----------------------------------------------------+------------------------+
| judul_buku                                         | maximum_pembelian_buku |
+----------------------------------------------------+------------------------+
| Building DMZs for Enterprise Networks              |                     10 |
| Computer Networking Illuminated                    |                      5 |
| Principles and Practices of Interconnection Networ |                     11 |
| Virtual Private Networks                           |                      4 |
+----------------------------------------------------+------------------------+
4 rows in set (0.000 sec)
```

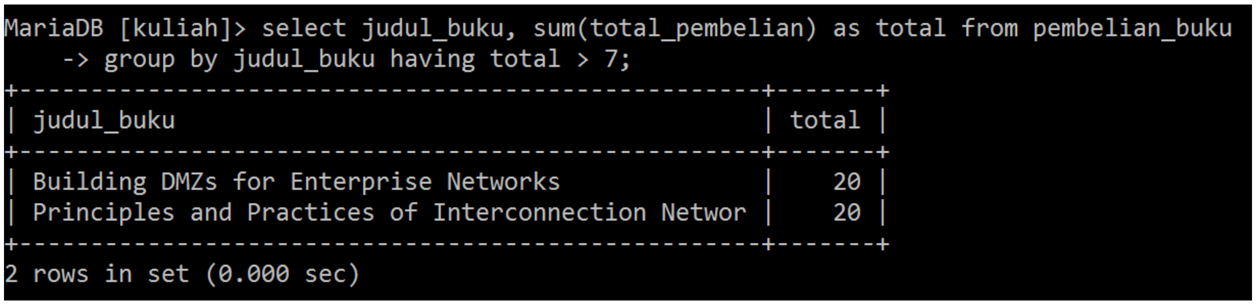
HAVING

- HAVING biasanya digunakan bersamaan dengan GROUP BY.
- HAVING sama dengan WHERE untuk GROUP BY.
- Fungsinya adalah untuk memberikan kondisi tertentu pada nilai hasil dari pengelompokan di GROUP BY.
- Syntax: SELECT nama_kolom1, … , nama_kolom_n, aggregate_function (nama_kolom) FROM nama_tabel WHERE kondisi GROUP BY nama_kolom HAVING kondisi;

Masih memakai database pembelian_buku.

Contoh Penggunaan HAVING (1)

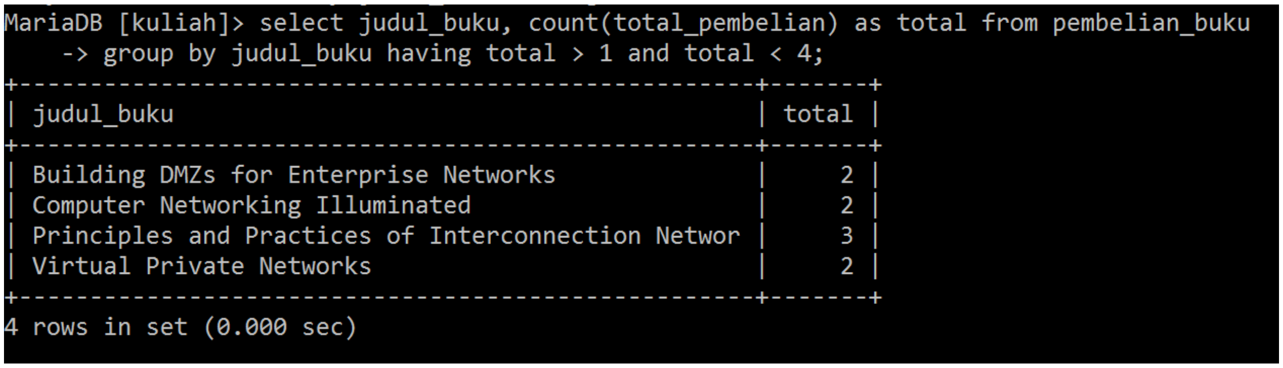
- Menampilkan data buku yang jumlah pembeliannya adalah lebih dari 7.
- Syntax: SELECT judul_buku, SUM(total_pembelian) AS total FROM pembelian_buku GROUP BY judul_buku HAVING total > 7;



```
MariaDB [kuliah]> select judul_buku, sum(total_pembelian) as total from pembelian_buku
    -> group by judul_buku having total > 7;
+---------------------------------------------------+-------+
| judul_buku                                        | total |
+---------------------------------------------------+-------+
| Building DMZs for Enterprise Networks             |    20 |
| Principles and Practices of Interconnection Networ |    20 |
+---------------------------------------------------+-------+
2 rows in set (0.000 sec)
```

Contoh Penggunaan HAVING (2)

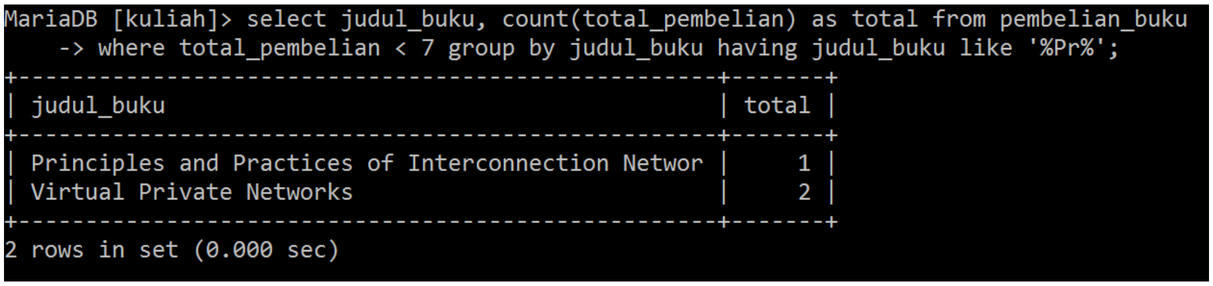
- Menampilkan data buku yang total transaksi penjualan perharinya adalah dalam rentang antara 2 – 3 buah buku.
- Syntax: SELECT judul buku, COUNT(total_pembelian) AS total FROM pembelian_buku GROUP BY judul_buku HAVING total > 1 AND total < 4;



```
MariaDB [kuliah]> select judul_buku, count(total_pembelian) as total from pembelian_buku
    -> group by judul_buku having total > 1 and total < 4;
+---------------------------------------------------+-------+
| judul_buku                                        | total |
+---------------------------------------------------+-------+
| Building DMZs for Enterprise Networks             |     2 |
| Computer Networking Illuminated                   |     2 |
| Principles and Practices of Interconnection Networ |     3 |
| Virtual Private Networks                          |     2 |
+---------------------------------------------------+-------+
4 rows in set (0.000 sec)
```

Contoh Penggunaan HAVING (3)

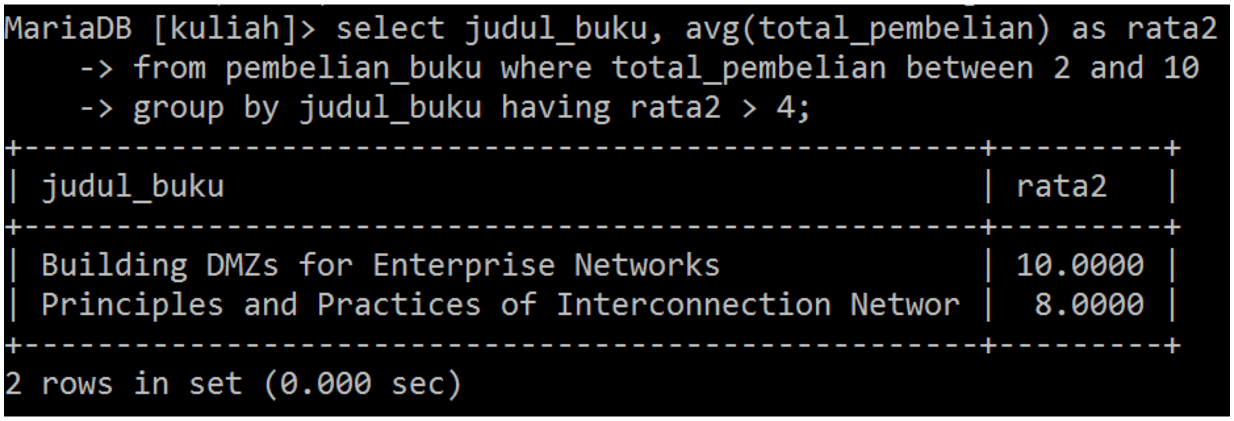
- Menampilkan total transaksi penjualan semua buku yang mengandung kata %Pr% yang total penjualan perharinya kurang dari 7 buah.
- Syntax: SELECT judul buku, COUNT(total_pembelian) AS total FROM pembelian_buku WHERE total_pembelian < 7 GROUP BY judul_buku HAVING judul_buku LIKE %Pr%;



```
MariaDB [kuliah]> select judul_buku, count(total_pembelian) as total from pembelian_buku
    -> where total_pembelian < 7 group by judul_buku having judul_buku like '%Pr%';
+-----------------------------------------------+-------+
| judul_buku                                    | total |
+-----------------------------------------------+-------+
| Principles and Practices of Interconnection Networ |     1 |
| Virtual Private Networks                      |     2 |
+-----------------------------------------------+-------+
2 rows in set (0.000 sec)
```

Contoh Penggunaan HAVING (4)

- Menampilkan data buku yang rata-rata terjual lebih dari 4 buah perhari, dan hanya buku yang terjual 2 – 10 buah perhari yang dihitung rata-ratanya.
- Syntax: SELECT judul buku, AVG(total_pembelian) AS rata-rata FROM pembelian_buku WHERE total_pembelian BETWEEN 2 AND 10 GROUP BY judul_buku HAVING total > 4;



```
MariaDB [kuliah]> select judul_buku, avg(total_pembelian) as rata2
    -> from pembelian_buku where total_pembelian between 2 and 10
    -> group by judul_buku having rata2 > 4;
+----------------------------------------------------+---------+
| judul_buku                                         | rata2   |
+----------------------------------------------------+---------+
| Building DMZs for Enterprise Networks              | 10.0000 |
| Principles and Practices of Interconnection Networ |  8.0000 |
+----------------------------------------------------+---------+
2 rows in set (0.000 sec)
```

SUBQUERY

1. Subquery untuk Menampilkan Data Berdasarkan Nilai Maksimum / Minimum

Contoh:
Tampilkan buku dengan total pembelian tertinggi dari tabel pembelian_buku.

```
SELECT judul_buku, total_pembelian
FROM pembelian_buku
WHERE total_pembelian = (SELECT MAX(total_pembelian) FROM pembelian_buku);
```

Konsep: Subquery mencari nilai maksimum dulu, lalu query utama menampilkan datanya.

---

2. Subquery untuk Menampilkan Data yang Lebih Besar dari Rata-rata

Contoh:
Tampilkan buku yang jumlah pembeliannya di atas rata-rata semua pembelian.

```
SELECT judul_buku, total_pembelian
FROM pembelian_buku
WHERE total_pembelian > (SELECT AVG(total_pembelian) FROM pembelian_buku);
```

Konsep: Subquery menghitung rata-rata, lalu digunakan sebagai pembanding.

---

3. Subquery di Dalam Klausa IN

Contoh:
Tampilkan buku yang pernah dibeli dan datanya juga ada di tabel `stock_buku`.

```
SELECT judul_buku
FROM pembelian_buku
WHERE judul_buku IN (SELECT judul_buku FROM stock_buku);
```

Konsep: Subquery menghasilkan daftar judul yang dibandingkan dengan tabel lain.

---

4. Subquery di Dalam Klausa FROM (Subquery Sebagai Tabel Sementara)

Contoh:
Hitung rata-rata pembelian setiap buku, lalu tampilkan hasilnya dengan alias.

```
SELECT judul_buku, AVG(total_pembelian) AS rata_rata
FROM (
    SELECT judul_buku, total_pembelian
    FROM pembelian_buku
) AS temp
GROUP BY judul_buku;
```

Konsep: Subquery digunakan sebagai tabel sementara untuk analisis lanjutan.

5. Subquery dengan `EXISTS`

Contoh:
Tampilkan buku yang punya stok di tabel `stock_buku`.

```
SELECT judul_buku
FROM pembelian_buku pb
WHERE EXISTS (
    SELECT * FROM stock_buku sb
    WHERE sb.judul_buku = pb.judul_buku
);
```

Konsep: Mengecek apakah data dari tabel pertama juga ada di tabel kedua.

---

6. Subquery untuk Menyaring Data Berdasarkan Tanggal Terbaru

Contoh:
Tampilkan pembelian dengan tanggal paling baru.

```
SELECT *
FROM pembelian_buku
WHERE tanggal_pembelian = (SELECT MAX(tanggal_pembelian) FROM pembelian_buku);
```

---

7. Subquery Bersarang (Nested Subquery)

Contoh:
Cari buku dengan total pembelian lebih besar dari rata-rata pembelian buku yang stoknya tidak kosong.

```
SELECT judul_buku, total_pembelian
FROM pembelian_buku
WHERE total_pembelian > (
    SELECT AVG(total_pembelian)
    FROM pembelian_buku
    WHERE judul_buku IN (
        SELECT judul_buku FROM stock_buku WHERE jumlah_stock IS NOT NULL
    )
);
```

Konsep: Subquery di dalam subquery untuk analisis yang lebih kompleks.

JOINS

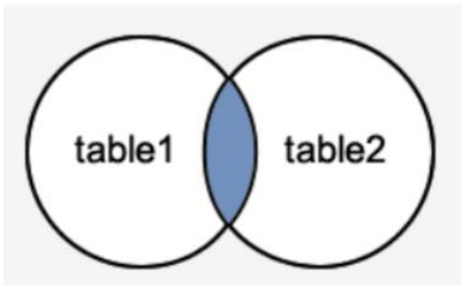
Joins digunakan bersamaan dengan perintah SELECT untuk mengambil data dari dua tabel atau lebih.

Tipe Joins:

- Inner Join
- Left Join
- Right Join

Inner Join

Data yang ditampilkan pada inner join adalah semua baris data yang memiliki kecocokan data pada kolom antar tabel.



Syntax: SELECT kolom_1, … , kolom_n FROM nama_tabel_1 INNER JOIN nama_tabel_2 ON kondisi_1, INNER JOIN nama_tabel_3 ON kondisi_2, … ;

Contoh database :

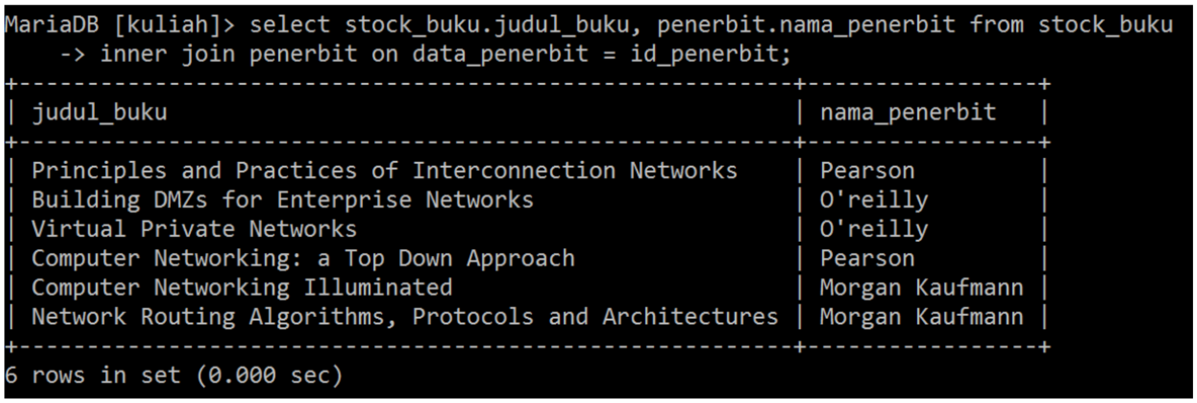
```
CREATE TABLE penerbit (

    id_penerbit INT(11) NOT NULL AUTO_INCREMENT,

    nama_penerbit VARCHAR(25),

    kota VARCHAR(25),

    PRIMARY KEY (id_penerbit)

);
```

```
CREATE TABLE stock_buku (

    id_buku INT(11) NOT NULL AUTO_INCREMENT,

    judul_buku VARCHAR(60),

    data_penerbit INT(11),

    jumlah_stock INT(11),

    PRIMARY KEY (id_buku),

    FOREIGN KEY (data_penerbit) REFERENCES penerbit(id_penerbit)

);

INSERT INTO penerbit (id_penerbit, nama_penerbit, kota) VALUES

(1001, 'Pearson', 'London'),

(1002, 'O\'reilly', 'California'),

(1003, 'Morgan Kaufmann', 'Massachusetts');


INSERT INTO stock_buku (id_buku, judul_buku, data_penerbit, jumlah_stock) VALUES

(1, 'Principles and Practices of Interconnection Networks', 1001, 10),

(2, 'Building DMZs for Enterprise Networks', 1002, 20),

(3, 'Virtual Private Networks', 1002, 11),

(4, 'Computer Networking: a Top Down Approach', 1001, 4),

(5, 'Computer Networking Illuminated', 1003, 5),

(6, 'Network Routing Algorithms, Protocols and Architectures', 1003, 7);
```

Contoh Penggunaan Inner Join (1)

Menampilkan nama penerbit untuk setiap judul buku.

Syntax: SELECT stock_buku.judul_buku, penerbit.nama_penerbit FROM stock_buku INNER JOIN penerbit ON data_penerbit = id_penerbit;

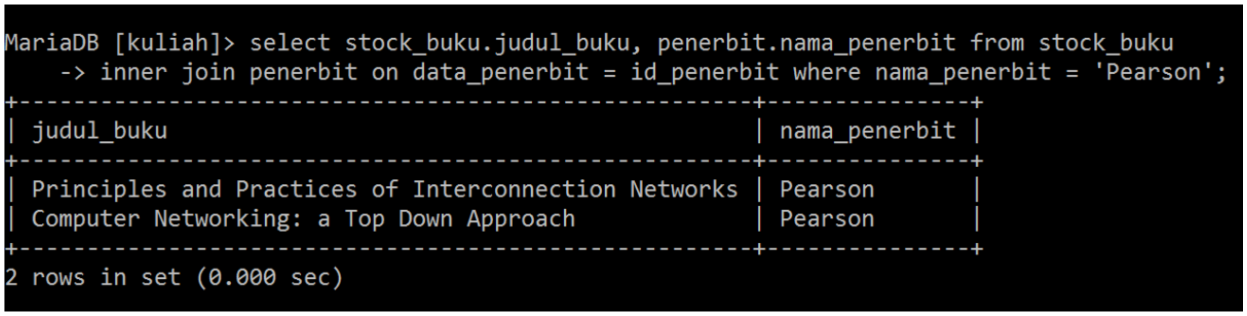


```
MariaDB [kuliah]> select stock_buku.judul_buku, penerbit.nama_penerbit from stock_buku
    -> inner join penerbit on data_penerbit = id_penerbit;
+-------------------------------------------------------+-----------------+
| judul_buku                                            | nama_penerbit   |
+-------------------------------------------------------+-----------------+
| Principles and Practices of Interconnection Networks  | Pearson         |
| Building DMZs for Enterprise Networks                 | O'reilly        |
| Virtual Private Networks                              | O'reilly        |
| Computer Networking: a Top Down Approach              | Pearson         |
| Computer Networking Illuminated                       | Morgan Kaufmann |
| Network Routing Algorithms, Protocols and Architectures | Morgan Kaufmann |
+-------------------------------------------------------+-----------------+
6 rows in set (0.000 sec)
```

Contoh Penggunaan Inner Join (2)

Menampilkan semua nama buku yang diterbitkan oleh Pearson.

Syntax: SELECT stock_buku.judul_buku, penerbit.nama_penerbit FROM stock_buku INNER JOIN penerbit ON data_penerbit = id_penerbit WHERE nama_penerbit = ‘Pearson’;

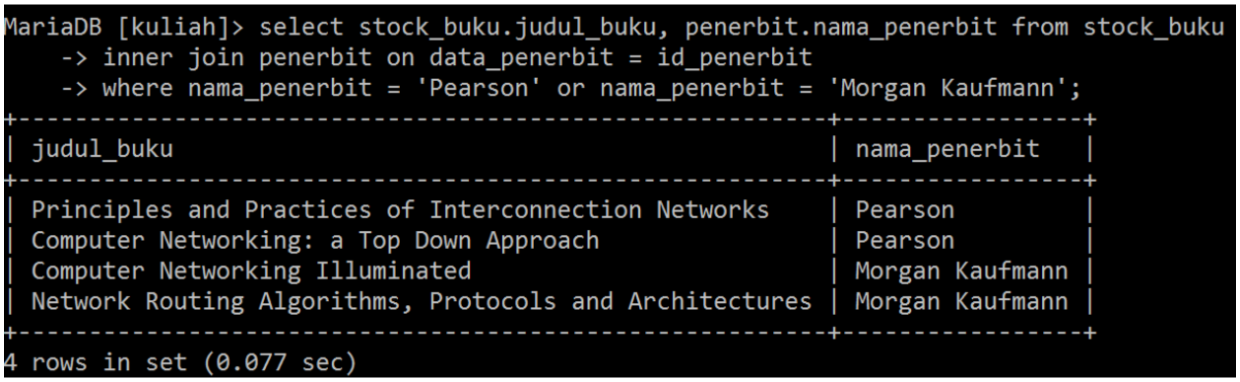


```
MariaDB [kuliah]> select stock_buku.judul_buku, penerbit.nama_penerbit from stock_buku
    -> inner join penerbit on data_penerbit = id_penerbit where nama_penerbit = 'Pearson';
+------------------------------------------------------+---------------+
| judul_buku                                           | nama_penerbit |
+------------------------------------------------------+---------------+
| Principles and Practices of Interconnection Networks | Pearson       |
| Computer Networking: a Top Down Approach             | Pearson       |
+------------------------------------------------------+---------------+
2 rows in set (0.000 sec)
```

Contoh Penggunaan Inner Join (3)

Menampilkan semua nama buku yang diterbitkan oleh Pearson atau Morgan Kaufmann.

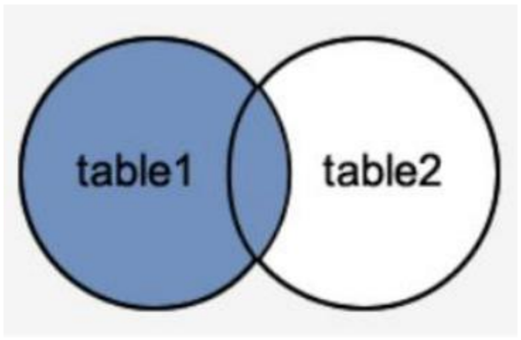
Syntax: SELECT stock_buku.judul_buku, penerbit.nama_penerbit FROM stock_buku INNER JOIN penerbit ON data_penerbit = id_penerbit WHERE nama_penerbit = ‘Pearson’ OR nama_penerbit = ‘Morgan Kaufmann’;



```
MariaDB [kuliah]> select stock_buku.judul_buku, penerbit.nama_penerbit from stock_buku
    -> inner join penerbit on data_penerbit = id_penerbit
    -> where nama_penerbit = 'Pearson' or nama_penerbit = 'Morgan Kaufmann';
+---------------------------------------------------------+-----------------+
| judul_buku                                              | nama_penerbit   |
+---------------------------------------------------------+-----------------+
| Principles and Practices of Interconnection Networks    | Pearson         |
| Computer Networking: a Top Down Approach                | Pearson         |
| Computer Networking Illuminated                         | Morgan Kaufmann |
| Network Routing Algorithms, Protocols and Architectures | Morgan Kaufmann |
+---------------------------------------------------------+-----------------+
4 rows in set (0.077 sec)
```

Left Join

Data yang ditampilkan pada left join adalah semua baris data yang ada pada tabel sebelah kiri (tabel 1), dan semua baris data pada tabel sebelah kanan (tabel 2) yang cocok dengan dengan tabel 1. Jika ada ketidak cocokan data antara tabel 1 dengan tabel 2, maka data tabel 2 akan bernilai null.



Syntax: SELECT kolom_1, … , kolom_n FROM nama_tabel_1 LEFT JOIN nama_tabel_2 ON kondisi_1, LEFT JOIN nama_tabel_3 ON kondisi_2, … ;

Contoh database (masih sama, hanya akan ada perbedaan di stock buku) :

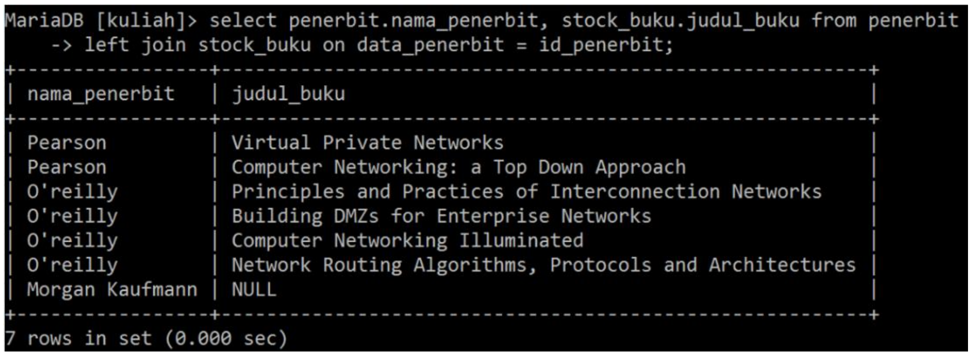
Jalankan ini,

```
UPDATE stock_buku

SET data_penerbit = CASE id_buku

    WHEN 1 THEN 1002

    WHEN 2 THEN 1002

    WHEN 3 THEN 1001

    WHEN 4 THEN 1001

    WHEN 5 THEN 1002

    WHEN 6 THEN 1002

END;
```

Contoh Penggunaan Left Join

Menampilkan nama penerbit buku yang stok bukunya masih tersedia.

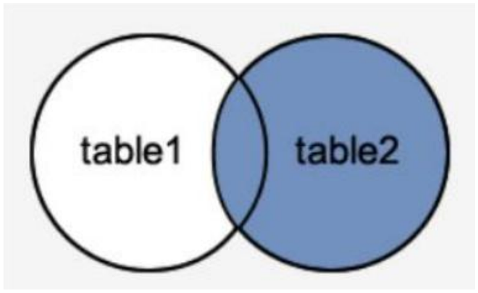
Syntax: SELECT penerbit.nama_penerbit, stock_buku.judul_buku FROM penerbit LEFT JOIN penerbit ON data_penerbit = id_penerbit;



```
MariaDB [kuliah]> select penerbit.nama_penerbit, stock_buku.judul_buku from penerbit
    -> left join stock_buku on data_penerbit = id_penerbit;
+----------------+-------------------------------------------------------+
| nama_penerbit  | judul_buku                                            |
+----------------+-------------------------------------------------------+
| Pearson        | Virtual Private Networks                              |
| Pearson        | Computer Networking: a Top Down Approach              |
| O'reilly       | Principles and Practices of Interconnection Networks  |
| O'reilly       | Building DMZs for Enterprise Networks                 |
| O'reilly       | Computer Networking Illuminated                       |
| O'reilly       | Network Routing Algorithms, Protocols and Architectures |
| Morgan Kaufmann | NULL                                                 |
+----------------+-------------------------------------------------------+
7 rows in set (0.000 sec)
```

Right Join

Data yang ditampilkan pada right join adalah semua baris data yang ada pada tabel sebelah kanan (tabel 2), dan semua baris data pada tabel sebelah kiri (tabel 1) yang cocok dengan dengan tabel 2. Jika ada ketidak cocokan data antara tabel 2 dengan tabel 1, maka data tabel 1 akan bernilai null

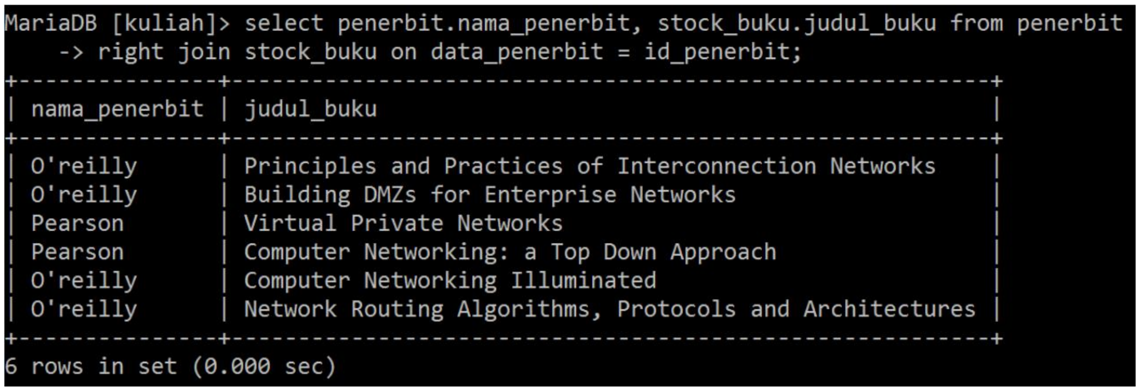


Syntax: SELECT kolom_1, … , kolom_n FROM nama_tabel_1 RIGHT JOIN nama_tabel_2 ON kondisi_1, RIGHT JOIN nama_tabel_3 ON kondisi_2, … ;

Contoh Penggunaan Right Join

Menampilkan nama penerbit buku yang stok bukunya masih tersedia

Syntax: SELECT penerbit.nama_penerbit, stock_buku.judul_buku FROM penerbit RIGHT JOIN penerbit ON data_penerbit = id_penerbit;



```
MariaDB [kuliah]> select penerbit.nama_penerbit, stock_buku.judul_buku from penerbit
    -> right join stock_buku on data_penerbit = id_penerbit;
+---------------+---------------------------------------------------------+
| nama_penerbit | judul_buku                                              |
+---------------+---------------------------------------------------------+
| O'reilly      | Principles and Practices of Interconnection Networks    |
| O'reilly      | Building DMZs for Enterprise Networks                   |
| Pearson       | Virtual Private Networks                                |
| Pearson       | Computer Networking: a Top Down Approach                |
| O'reilly      | Computer Networking Illuminated                         |
| O'reilly      | Network Routing Algorithms, Protocols and Architectures |
+---------------+---------------------------------------------------------+
6 rows in set (0.000 sec)
```

Trigger & Stored Procedure

KONSEP TRIGGER

Trigger merupakan suatu syntax pada SQL yang akan dieksekusi secara otomatis jika ada suatu kejadian tertentu pada database

Kejadian yang dapat memicu eksekusi Trigger meliputi INSERT, UPDATE, atau DELETE.

Tipe-tipe Trigger:

1. Before Trigger : trigger dieksekusi sebelum perubahan data ditampilkan.
2. After Trigger : trigger dieksekusi setelah perubahan data ditampilkan.

SYNTAX TRIGGER

- Pembuatan Trigger

```
CREATE TRIGGER nama_trigger

(AFTER | BEFORE) (INSERT | UPDATE | DELETE)

 ON nama_tabel FOR EACH ROW

 BEGIN

   --tuliskan kode SQL di sini

 END;
```

- Modifiers: OLD (data lama) dan NEW (data baru)
  - INSERT: NEW
  - UPDATE: OLD dan NEW
  - DELETE: OLD

- Hapus Trigger

```
Drop Trigger nama_trigger;
```

Buat table seperti ini didalam database :

Tabel stock barang :

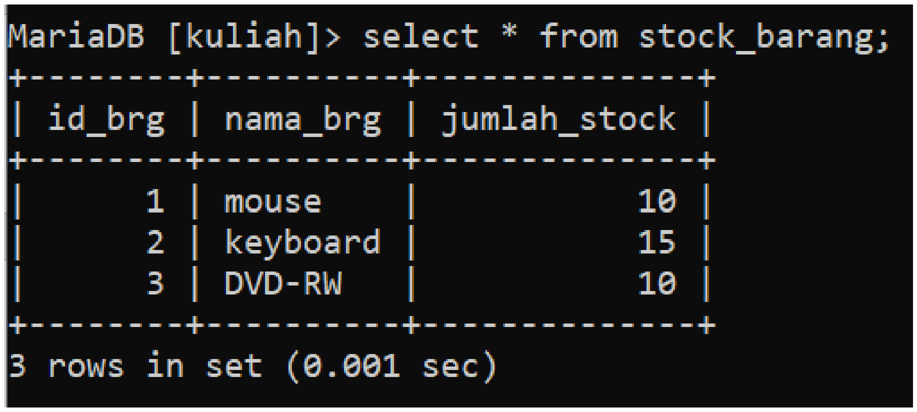
```
CREATE TABLE stock_barang (

    id_brg INT NOT NULL PRIMARY KEY AUTO_INCREMENT,

    nama_brg VARCHAR(25),

    jumlah_stock INT

);



INSERT INTO stock_barang (nama_brg, jumlah_stock) VALUES

('mouse', 10),

('keyboard', 15),

('DVD-RW', 10);
```

Tabel Pembelian :

```
CREATE TABLE pembelian (

    id_pembelian INT NOT NULL PRIMARY KEY AUTO_INCREMENT,

    id_brg INT NOT NULL,

    jml_pembelian INT NOT NULL,

    FOREIGN KEY (id_brg) REFERENCES stock_barang(id_brg)

);
```

Contoh Trigger :

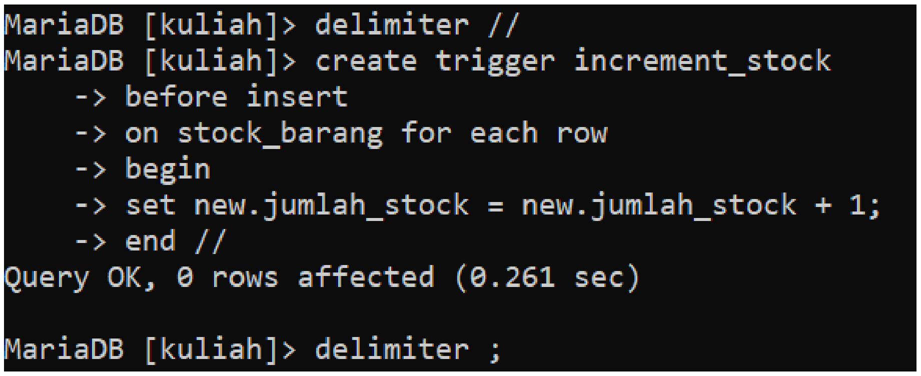
1. Buat Trigger untuk menambahkan jumlah_stock barang sebanyak 1 setiap ada penambahan data baru.



```
MariaDB [kuliah]> select * from stock_barang;
+--------+----------+--------------+
| id_brg | nama_brg | jumlah_stock |
+--------+----------+--------------+
|      1 | mouse    |           10 |
|      2 | keyboard |           15 |
|      3 | DVD-RW   |           10 |
+--------+----------+--------------+
3 rows in set (0.001 sec)
```

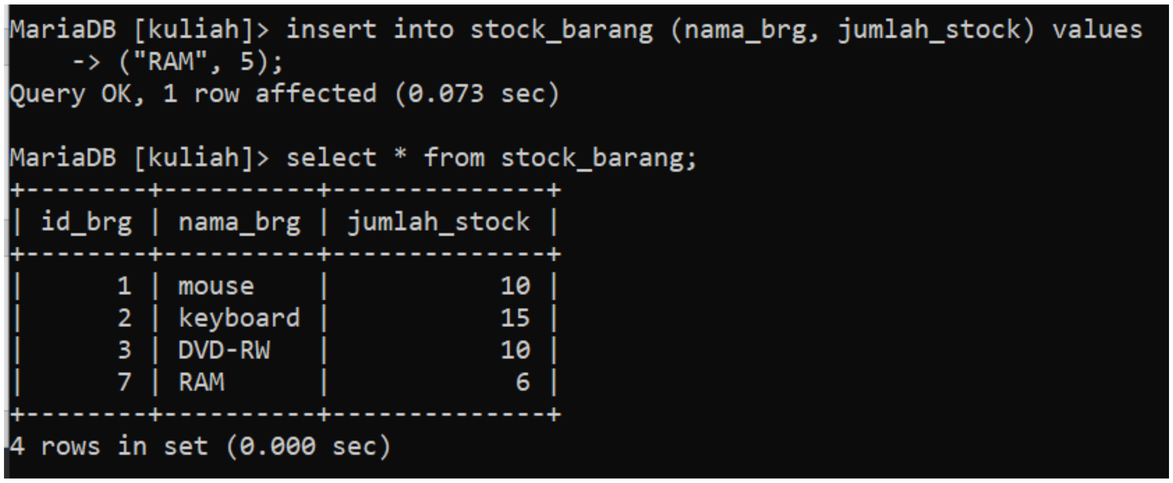
Contoh trigger (lanjutan) :

Pembuatan Trigger :



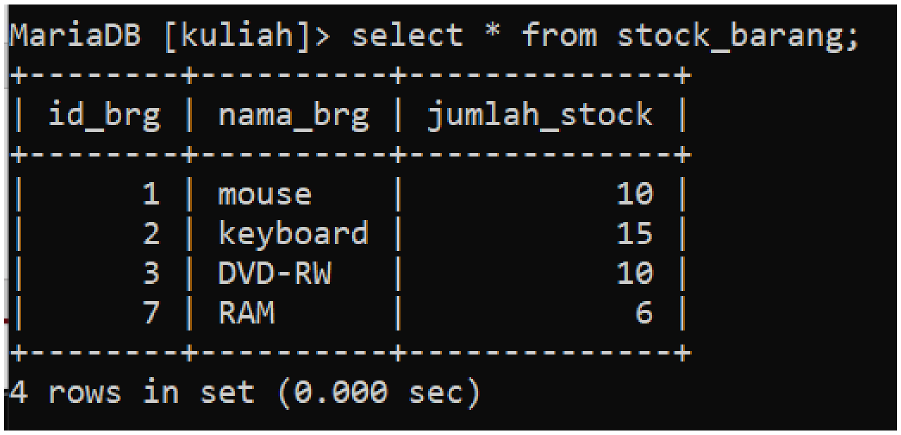
```
MariaDB [kuliah]> delimiter //
MariaDB [kuliah]> create trigger increment_stock
    -> before insert
    -> on stock_barang for each row
    -> begin
    -> set new.jumlah_stock = new.jumlah_stock + 1;
    -> end //
Query OK, 0 rows affected (0.261 sec)

MariaDB [kuliah]> delimiter ;
```

Insert stock barang baru :



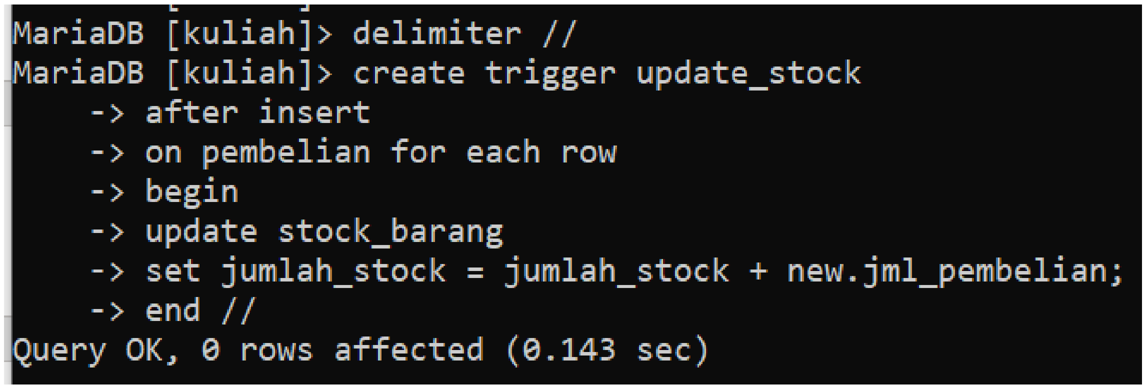
```
MariaDB [kuliah]> insert into stock_barang (nama_brg, jumlah_stock) values
    -> ("RAM", 5);
Query OK, 1 row affected (0.073 sec)

MariaDB [kuliah]> select * from stock_barang;
+--------+----------+--------------+
| id_brg | nama_brg | jumlah_stock |
+--------+----------+--------------+
|      1 | mouse    |           10 |
|      2 | keyboard |           15 |
|      3 | DVD-RW   |           10 |
|      7 | RAM      |            6 |
+--------+----------+--------------+
4 rows in set (0.000 sec)
```

2. Buat Trigger untuk menambahkan jumlah stock barang sebanyak pembelian barang



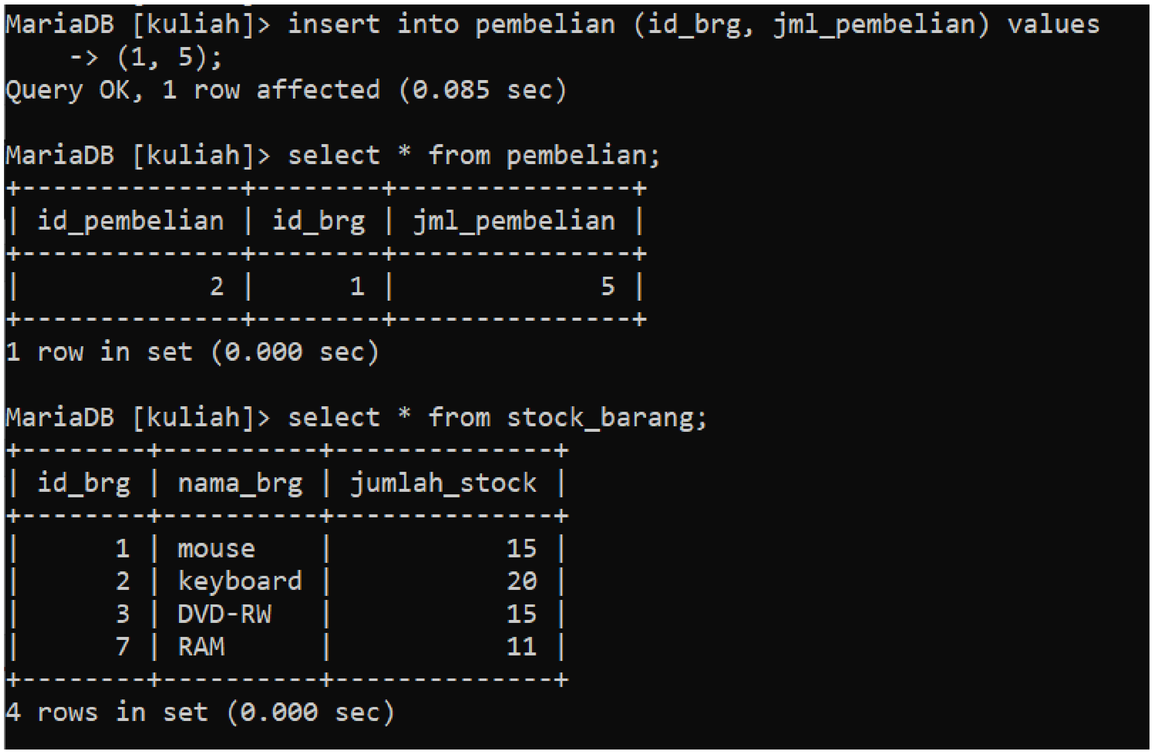
```
MariaDB [kuliah]> select * from stock_barang;
+--------+----------+--------------+
| id_brg | nama_brg | jumlah_stock |
+--------+----------+--------------+
|      1 | mouse    |           10 |
|      2 | keyboard |           15 |
|      3 | DVD-RW   |           10 |
|      7 | RAM      |            6 |
+--------+----------+--------------+
4 rows in set (0.000 sec)
```

Pembuatan trigger :



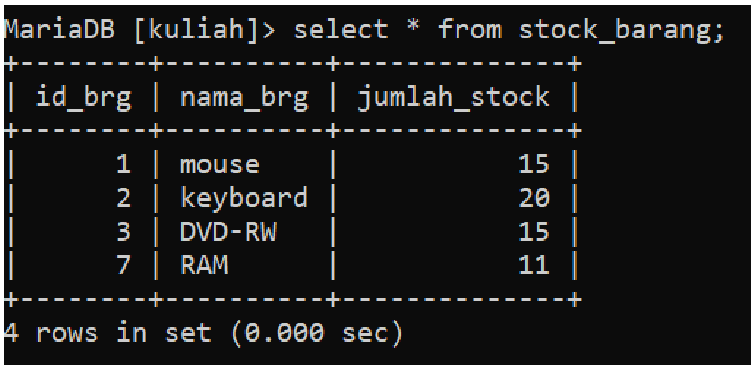
```
MariaDB [kuliah]> delimiter //
MariaDB [kuliah]> create trigger update_stock
    -> after insert
    -> on pembelian for each row
    -> begin
    -> update stock_barang
    -> set jumlah_stock = jumlah_stock + new.jml_pembelian;
    -> end //
Query OK, 0 rows affected (0.143 sec)
```

Insert pembelian barang baru :

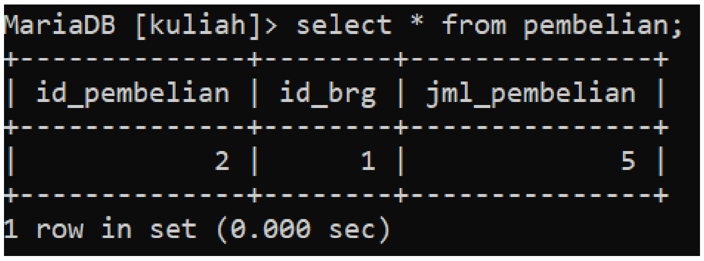


```
MariaDB [kuliah]> insert into pembelian (id_brg, jml_pembelian) values
    -> (1, 5);
Query OK, 1 row affected (0.085 sec)

MariaDB [kuliah]> select * from pembelian;
+--------------+--------+---------------+
| id_pembelian | id_brg | jml_pembelian |
+--------------+--------+---------------+
|            2 |      1 |             5 |
+--------------+--------+---------------+
1 row in set (0.000 sec)

MariaDB [kuliah]> select * from stock_barang;
+--------+----------+--------------+
| id_brg | nama_brg | jumlah_stock |
+--------+----------+--------------+
|      1 | mouse    |           15 |
|      2 | keyboard |           20 |
|      3 | DVD-RW   |           15 |
|      7 | RAM      |           11 |
+--------+----------+--------------+
4 rows in set (0.000 sec)
```

3. Buat Trigger untuk mengurangi jumlah stock barang jika dilakukan penghapusan data pembelian barang.

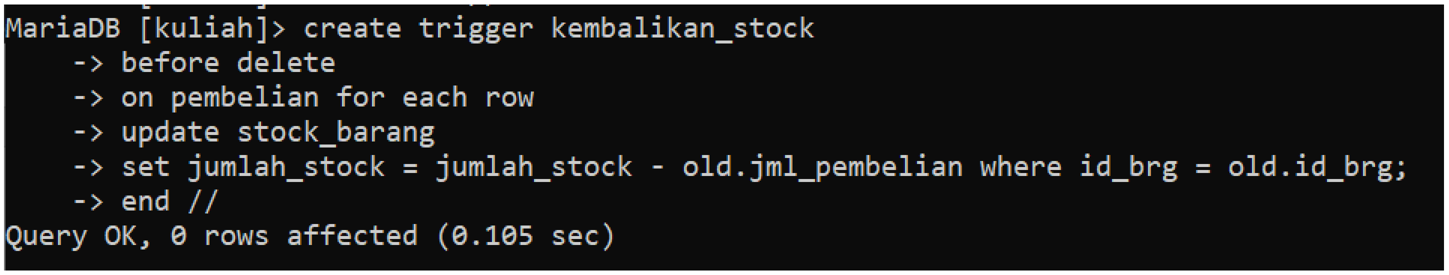


```
MariaDB [kuliah]> select * from stock_barang;
+--------+----------+--------------+
| id_brg | nama_brg | jumlah_stock |
+--------+----------+--------------+
|      1 | mouse    |           15 |
|      2 | keyboard |           20 |
|      3 | DVD-RW   |           15 |
|      7 | RAM      |           11 |
+--------+----------+--------------+
4 rows in set (0.000 sec)
```



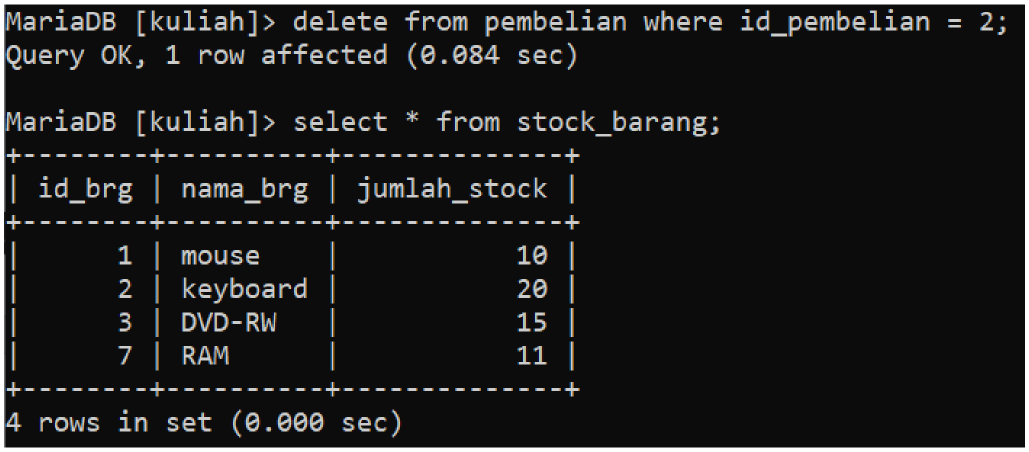
```
MariaDB [kuliah]> select * from pembelian;
+--------------+--------+---------------+
| id_pembelian | id_brg | jml_pembelian |
+--------------+--------+---------------+
|            2 |      1 |             5 |
+--------------+--------+---------------+
1 row in set (0.000 sec)
```

Pembuatan trigger :



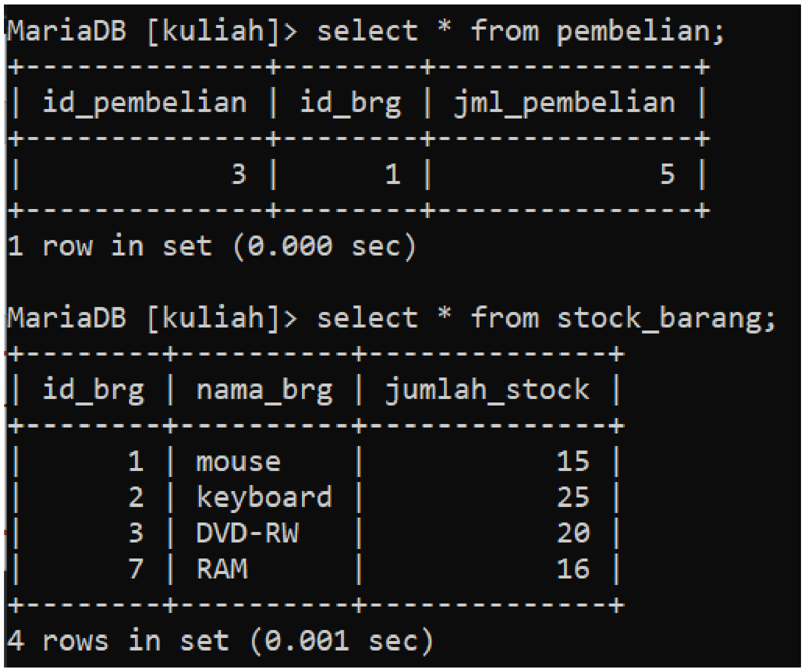
```
MariaDB [kuliah]> create trigger kembalikan_stock
    -> before delete
    -> on pembelian for each row
    -> update stock_barang
    -> set jumlah_stock = jumlah_stock - old.jml_pembelian where id_brg = old.id_brg;
    -> end //
Query OK, 0 rows affected (0.105 sec)
```

Hapus data pembelian :



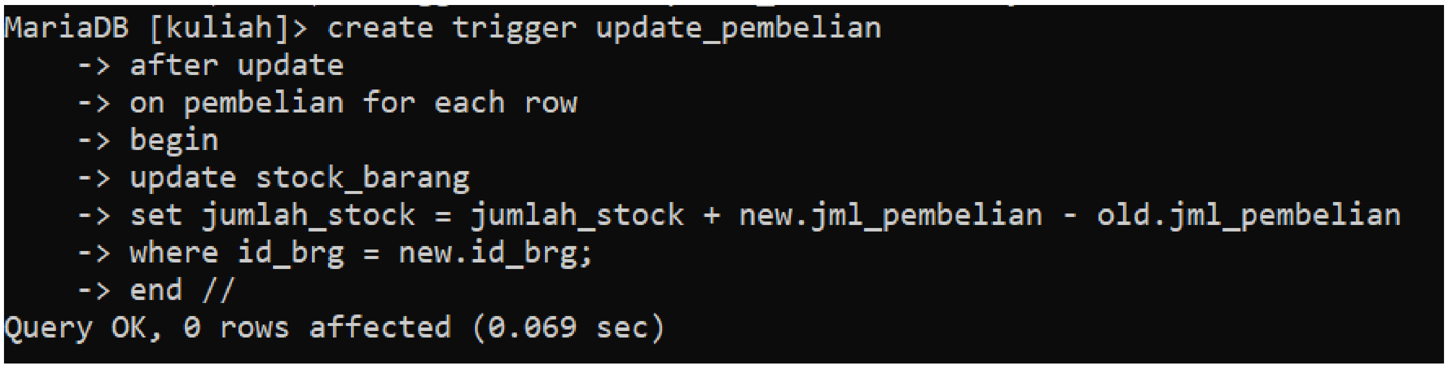
```
MariaDB [kuliah]> delete from pembelian where id_pembelian = 2;
Query OK, 1 row affected (0.084 sec)

MariaDB [kuliah]> select * from stock_barang;
+--------+----------+--------------+
| id_brg | nama_brg | jumlah_stock |
+--------+----------+--------------+
|      1 | mouse    |           10 |
|      2 | keyboard |           20 |
|      3 | DVD-RW   |           15 |
|      7 | RAM      |           11 |
+--------+----------+--------------+
4 rows in set (0.000 sec)
```

4. Buat Trigger untuk melakukan update jumlah stock barang jika terjadi perubahan data jumlah pembelian barang.



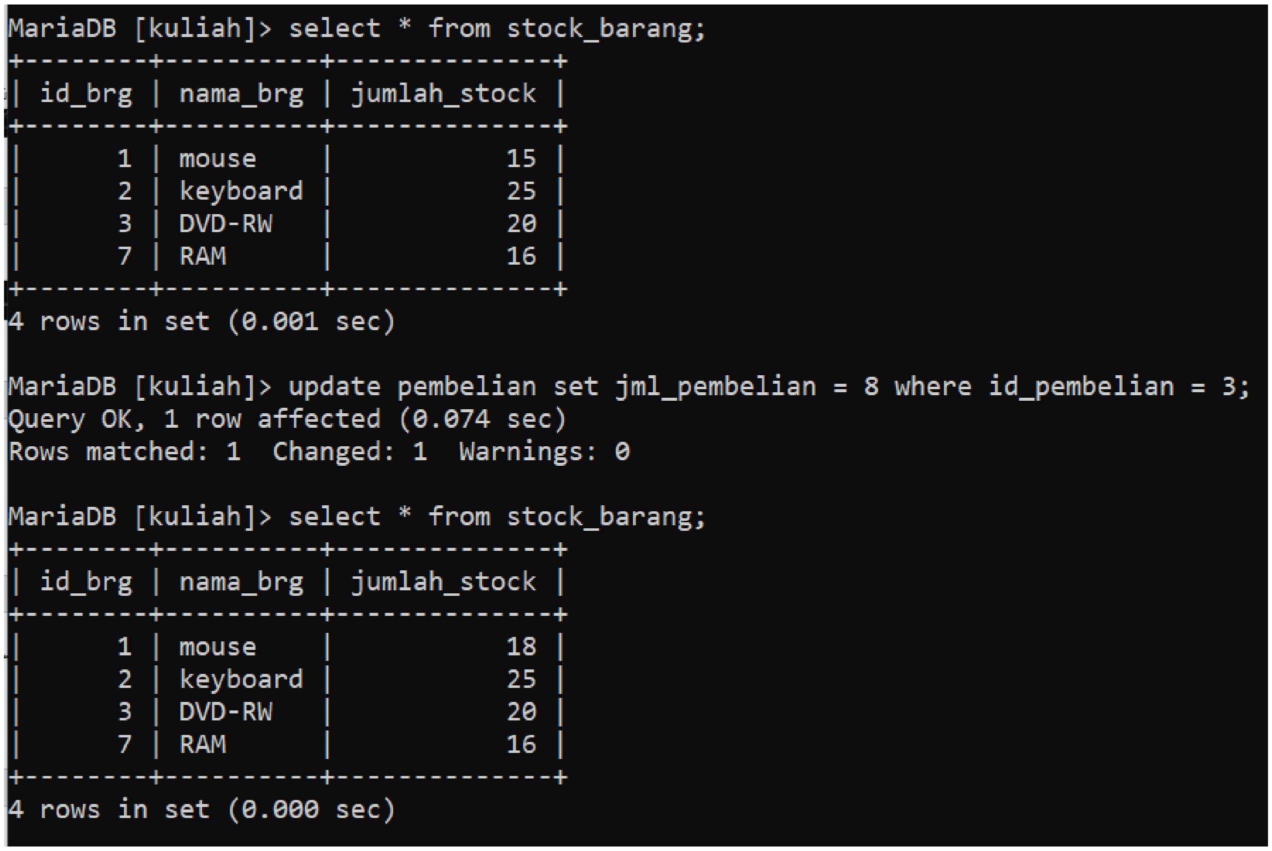
```
MariaDB [kuliah]> select * from pembelian;
+--------------+--------+---------------+
| id_pembelian | id_brg | jml_pembelian |
+--------------+--------+---------------+
|            3 |      1 |             5 |
+--------------+--------+---------------+
1 row in set (0.000 sec)

MariaDB [kuliah]> select * from stock_barang;
+--------+----------+--------------+
| id_brg | nama_brg | jumlah_stock |
+--------+----------+--------------+
|      1 | mouse    |           15 |
|      2 | keyboard |           25 |
|      3 | DVD-RW   |           20 |
|      7 | RAM      |           16 |
+--------+----------+--------------+
4 rows in set (0.001 sec)
```

Pembuatan trigger :



```
MariaDB [kuliah]> create trigger update_pembelian
    -> after update
    -> on pembelian for each row
    -> begin
    -> update stock_barang
    -> set jumlah_stock = jumlah_stock + new.jml_pembelian - old.jml_pembelian
    -> where id_brg = new.id_brg;
    -> end //
Query OK, 0 rows affected (0.069 sec)
```

Update data pembelian :



```
MariaDB [kuliah]> select * from stock_barang;
+--------+----------+--------------+
| id_brg | nama_brg | jumlah_stock |
+--------+----------+--------------+
|      1 | mouse    |           15 |
|      2 | keyboard |           25 |
|      3 | DVD-RW   |           20 |
|      7 | RAM      |           16 |
+--------+----------+--------------+
4 rows in set (0.001 sec)

MariaDB [kuliah]> update pembelian set jml_pembelian = 8 where id_pembelian = 3;
Query OK, 1 row affected (0.074 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [kuliah]> select * from stock_barang;
+--------+----------+--------------+
| id_brg | nama_brg | jumlah_stock |
+--------+----------+--------------+
|      1 | mouse    |           18 |
|      2 | keyboard |           25 |
|      3 | DVD-RW   |           20 |
|      7 | RAM      |           16 |
+--------+----------+--------------+
4 rows in set (0.000 sec)
```

STORED PROCEDURE

Konsep Stored Procedure :

Stored procedure merupakan kumpulan syntax SQL yang disimpan di dalam database, yang kemudian dieksekusi dengan cara memanggil nama stored procedure yang telah dibuat sebelumnya

Tujuan dari penggunaan stored procedure adalah untuk mempermudah pada saat beberapa aplikasi yang kita buat menggunakan database yang sama, sehingga jika ada perubahan syntax di dalam database, kita tidak perlu merubah syntax di semua aplikasi yang kita buat, melainkan cukup merubah syntax yang ada di dalam database.

Syntax Stored Procedure :

```
CREATE PROCEDURE nama_procedure([parameter[, ...]])

BEGIN

  tuliskan kode di sini

END;
```

Parameter:

1. **In**: pada saat melakukan pemanggilan stored procedure harus memasukkan nilai input dimana nilai inputan tersebut akan diproses di stored procedure
2. **Out**: pada saat melakukan pemanggilan stored procedure, maka nilai dari hasil proses yang ada di dalam stored procedure akan dikembalikan ke pemanggil.
3. **INOUT**: merupakankombinasi dari parameter IN dan OUT.

Memanggil stored procedure:

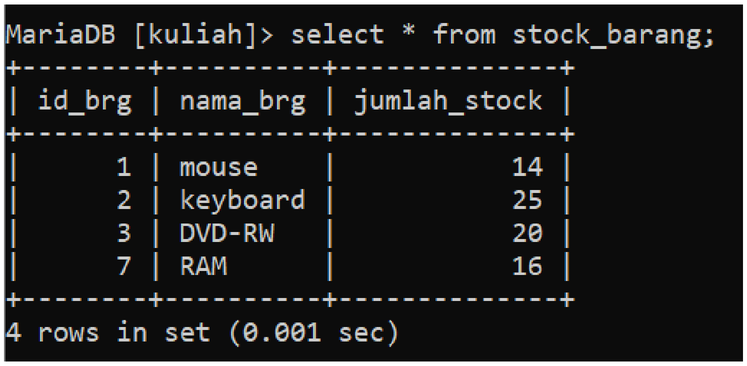
```
 CALL nama_procedure (parameter)
```

Menghapus stored procedure:

```
 DROP nama_procedure
```

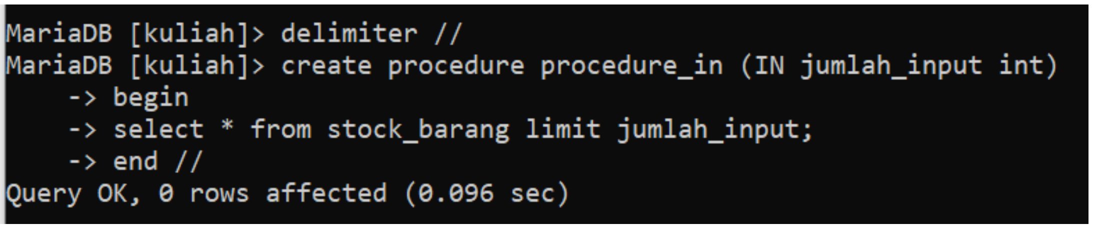
Contoh Stored Procedure Procedure IN

1. Membuat stored procedure untuk membatasi jumlah data yang ditampilkan.



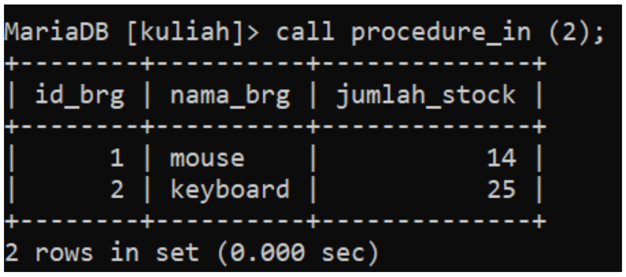
```
MariaDB [kuliah]> select * from stock_barang;
+--------+----------+--------------+
| id_brg | nama_brg | jumlah_stock |
+--------+----------+--------------+
|      1 | mouse    |           14 |
|      2 | keyboard |           25 |
|      3 | DVD-RW   |           20 |
|      7 | RAM      |           16 |
+--------+----------+--------------+
4 rows in set (0.001 sec)
```

Pembuatan Stored Procedure :



```
MariaDB [kuliah]> delimiter //
MariaDB [kuliah]> create procedure procedure_in (IN jumlah_input int)
    -> begin
    -> select * from stock_barang limit jumlah_input;
    -> end //
Query OK, 0 rows affected (0.096 sec)
```

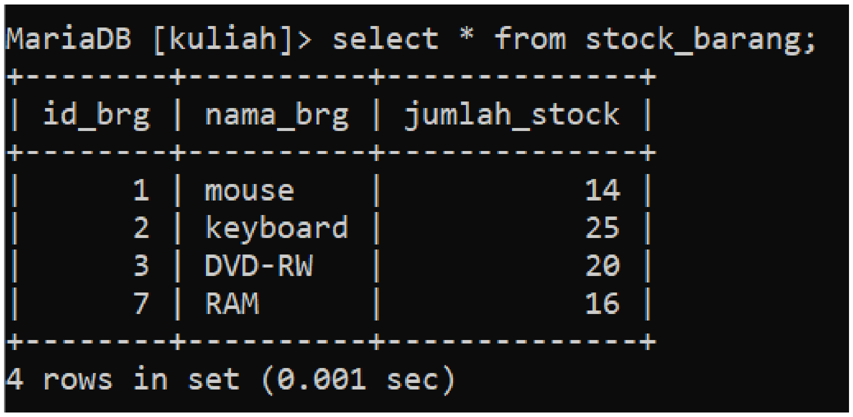
Memanggil Stored Procedure :



```
MariaDB [kuliah]> call procedure_in (2);
+--------+----------+--------------+
| id_brg | nama_brg | jumlah_stock |
+--------+----------+--------------+
|      1 | mouse    |           14 |
|      2 | keyboard |           25 |
+--------+----------+--------------+
2 rows in set (0.000 sec)
```

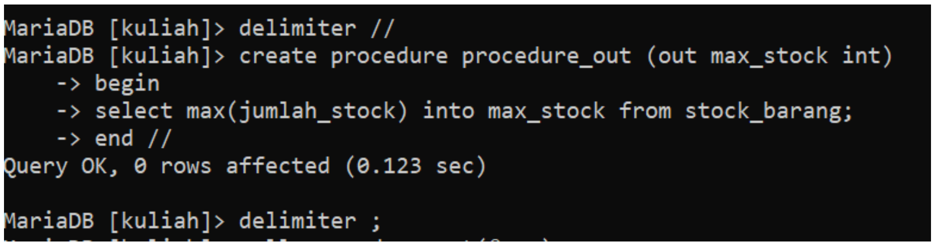
Contoh Stored Procedure Procedure OUT

2. Membuat stored procedure untuk menampilkan jumlah stock barang yang paling banyak.

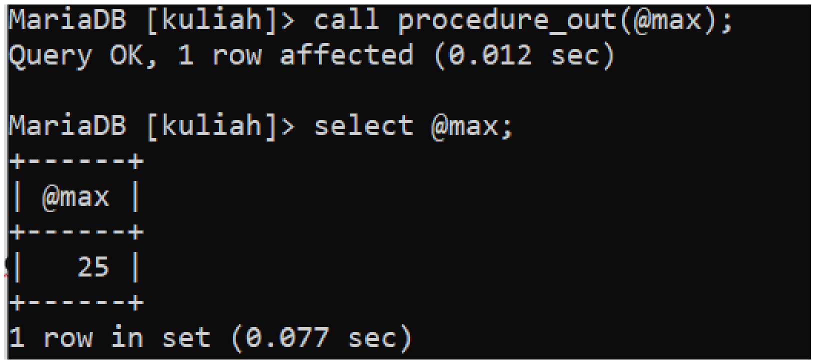


```
MariaDB [kuliah]> select * from stock_barang;
+--------+----------+--------------+
| id_brg | nama_brg | jumlah_stock |
+--------+----------+--------------+
|      1 | mouse    |           14 |
|      2 | keyboard |           25 |
|      3 | DVD-RW   |           20 |
|      7 | RAM      |           16 |
+--------+----------+--------------+
4 rows in set (0.001 sec)
```

Pembuatan Stored Procedure :

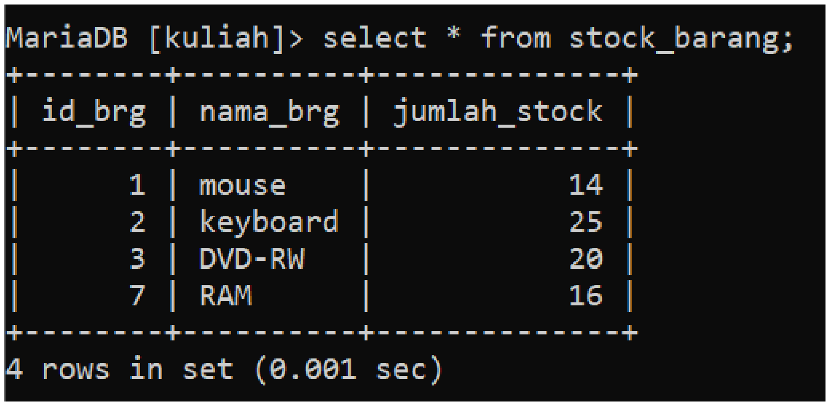

```
MariaDB [kuliah]> delimiter //
MariaDB [kuliah]> create procedure procedure_out (out max_stock int)
    -> begin
    -> select max(jumlah_stock) into max_stock from stock_barang;
    -> end //
Query OK, 0 rows affected (0.123 sec)

MariaDB [kuliah]> delimiter ;
```

Memanggil Stored Procedure :

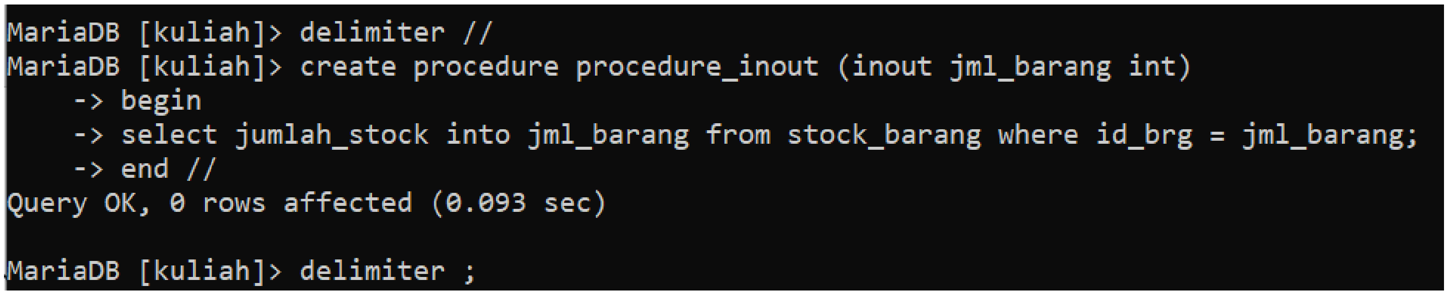

```
MariaDB [kuliah]> call procedure_out(@max);
Query OK, 1 row affected (0.012 sec)

MariaDB [kuliah]> select @max;
+------+
| @max |
+------+
|   25 |
+------+
1 row in set (0.077 sec)
```

Contoh Stored Procedure Procedure INOUT

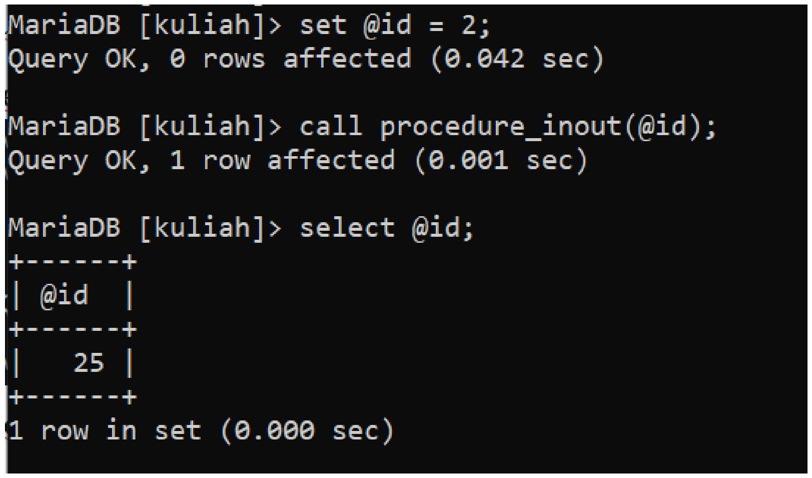
3. Membuat stored procedure untuk menampilkan jumlah stock barang untuk id_brg = 2.


```
MariaDB [kuliah]> select * from stock_barang;
+--------+----------+--------------+
| id_brg | nama_brg | jumlah_stock |
+--------+----------+--------------+
|      1 | mouse    |           14 |
|      2 | keyboard |           25 |
|      3 | DVD-RW   |           20 |
|      7 | RAM      |           16 |
+--------+----------+--------------+
4 rows in set (0.001 sec)
```

Pembuatan Stored Procedure :


```
MariaDB [kuliah]> delimiter //
MariaDB [kuliah]> create procedure procedure_inout (inout jml_barang int)
    -> begin
    -> select jumlah_stock into jml_barang from stock_barang where id_brg = jml_barang;
    -> end //
Query OK, 0 rows affected (0.093 sec)

MariaDB [kuliah]> delimiter ;
```

Memanggil Stored Procedure :



```
MariaDB [kuliah]> set @id = 2;
Query OK, 0 rows affected (0.042 sec)

MariaDB [kuliah]> call procedure_inout(@id);
Query OK, 1 row affected (0.001 sec)

MariaDB [kuliah]> select @id;
+------+
| @id  |
+------+
|   25 |
+------+
1 row in set (0.000 sec)
```