

OPERATOR DAN DML (DATA MANIPULATION LANGUAGE)

OBJEKTIF :

1. Mahasiswa mampu menggunakan operator AND dan OR.
2. Mahasiswa mampu menggunakan perintah dalam DML (*Data Manipulation Language*).

4.1 OPERATOR

Operator dalam SQL digunakan untuk menginstruksikan perintah agar melakukan sesuatu. Terdapat beberapa operator yaitu :

4.1.1 Operator AND

Operator `AND` adalah operator logika yang menggabungkan dua atau lebih ekspresi. Operator `AND` sering digunakan dalam klausa `WHERE` pada pernyataan `SELECT`, `UPDATE`, dan `DELETE` untuk membentuk suatu kondisi. Berikut bentuk umum penggunaan operator `AND` :

```
condition_1 AND condition_2
```

Penjelasan :

- `condition_1, condition_2` : Merupakan kondisi untuk menentukan data yang diinginkan

Operator `AND` memiliki tabel kebenaran untuk membedakan fungsi operator `AND` dengan operator `OR` :

	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

Melihat dari tabel kebenaran jika kedua data bernilai *True* maka perintah yang diminta akan dijalankan atau *True*. Jika kedua data *False* maka perintah yang diminta tidak akan di jalankan atau *False*. Jika satu data bernilai *True* dan satu data lainnya bernilai *False* maka perintah yang diminta tidak akan di jalankan atau *False*. Jika satu data bernilai *True* dan satu data lainnya bernilai *Null* maka hasil dari perintah yang diminta adalah *Null* atau kosong. Jika satu data bernilai *False* dan satu data lainnya bernilai *Null* maka hasil dari perintah yang diminta tidak ditampilkan *False*.

Tabel Pengguna :

	IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
▶	1	Sarah	081289987117	sarahayuh@gmail.com	Jl.Pondok 38	Jakarta
	2	Dyan	081267761355	dyantriandini@yahoo.com	Jl.Sari D35	Bandung
	3	Riza	081245677654	riza123@gmail.com	Jl.Angkasa	Bogor
	4	Ananda	081289891111	ananda@gmail.com	Jl.Markisa No.2	Bogor
	5	Viora	081122334465	viora@gmail.com	Jl.Wijaya	Jakarta
	6	Andika	082278905678	andika@gmail.com	Jl.Wijaya	NULL

Berikut contoh pemakaian operator `AND` pada tabel Pengguna :

- `SELECT * FROM Pengguna WHERE Alamat='Jl.Wijaya' AND Kota='Jakarta';`

Hasil :

	IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
▶	5	Viora	081122334465	viora@gmail.com	Jl.Wijaya	Jakarta

Dilihat dari tabel pengguna terdapat dua baris yang memiliki Alamat = 'Jl.Wijaya', yaitu NamaPengguna Viora dan Andika. Query di atas meminta untuk menampilkan Alamat='Jl.Wijaya' dan Kota='Jakarta'. Untuk NamaPengguna Andika berarti memiliki kondisi *True* (Alamat='Jl.Wijaya') dan *Null* (Kota=*Null*) maka dari itu dilihat dari tabel kebenaran *True* bertemu dengan *Null* (kosong) hasilnya *Null* (kosong) data Andika tidak ditampilkan. Jika melihat data NamaPengguna Viora dengan kondisi *True* (Alamat = 'Jl.Wijaya') dan *True* (Kota = 'Jakarta'), dilihat dari tabel kebenaran *True* bertemu *True* hasilnya adalah *True*. Oleh sebab itu data Viora ditampilkan.

- `SELECT * FROM Pengguna WHERE Alamat='Jl.Wijaya' AND Kota=NULL;`

Hasil :

	IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
•	NULL	NULL	NULL	NULL	NULL	NULL

Dilihat dari tabel pengguna terdapat dua baris yang memiliki Alamat = 'Jl.Wijaya', yaitu NamaPengguna Viora dan Andika. Query di atas meminta untuk menampilkan Alamat='Jl.Wijaya' dan Kota=*Null*. Untuk NamaPengguna Andika berarti memiliki kondisi *True* (Alamat='Jl.Wijaya') dan *Null* (Kota=*Null*) maka dari itu dilihat dari tabel kebenaran *True* bertemu dengan *Null* (kosong) hasilnya *Null* (kosong). Oleh sebab itu, data Andika tidak ditampilkan. Jika melihat data NamaPengguna Viora dengan kondisi *True* (Alamat = 'Jl.Wijaya') dan *False* (Kota = 'Jakarta'), dilihat dari tabel kebenaran *True* bertemu *False* hasilnya adalah *False*. Oleh sebab itu data Viora tidak ditampilkan.

- `SELECT * FROM Pengguna WHERE Alamat='Jl.Wijaya' AND Kota='Bogor';`

Hasil :

	IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
•	NULL	NULL	NULL	NULL	NULL	NULL

Dilihat dari tabel pengguna terdapat dua baris yang memiliki Alamat = 'Jl.Wijaya', yaitu NamaPengguna Viora dan Andika. Query yang di atas meminta untuk menampilkan Alamat='Jl.Wijaya' dan Kota='Bogor'. Untuk NamaPengguna Andika berarti memiliki kondisi *True* (Alamat='Jl.Wijaya') dan *Null* (Kota= *Null*) maka dari itu dilihat dari tabel kebenaran *True* bertemu dengan *Null* (kosong) hasilnya *Null* (kosong), data Andika tidak ditampilkan. Jika

melihat data NamaPengguna Viora dengan kondisi *True* (Alamat = 'Jl.Wijaya') dan *False* (Kota ='Jakarta'), dilihat dari tabel kebenaran *True* bertemu *False* hasilnya adalah *False*. Oleh sebab itu data Viora tidak ditampilkan.

4.1.2 Operator OR

Operator **OR** adalah operator logika yang menggabungkan dua atau lebih ekspresi. Operator **OR** sering digunakan dalam klausa **WHERE** pada pernyataan **SELECT**, **UPDATE**, dan **DELETE** untuk membentuk suatu kondisi. Pengertian awal operator **OR** mungkin mirip dengan operator **AND**, namun yang membedakan kedua operator tersebut adalah tabel kebenarannya. Berikut bentuk umum penggunaan operator **OR** :

```
condition_1 OR condition_2
```

Penjelasan :

- **condition_1, condition_2** : Merupakan kondisi untuk menentukan data yang diinginkan

Berikut tabel kebenaran dari operator **OR** :

	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

Terlihat dari tabel kebenaran jika kedua data bernilai *True* maka perintah yang diminta akan dijalankan atau *True*. Jika kedua data *False* maka perintah yang diminta tidak akan di jalankan atau *False*. Jika satu data bernilai *True* dan satu data lainnya bernilai *False* maka data akan ditampilkan atau *True*. Jika satu data bernilai *True* dan satu data lainnya bernilai *Null* maka hasil dari perintah yang diminta akan ditampilkan atau *True*. Jika satu data bernilai *False* dan satu data lainnya bernilai *Null* maka hasil dari perintah yang diminta *Null* atau kosong.

Tabel Pengguna

	IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
▶	1	Sarah	081289987117	sarahayuh@gmail.com	Jl.Pondok 38	Jakarta
	2	Dyan	081267761355	dyantriandini@yahoo.com	Jl.Sari D35	Bandung
	3	Riza	081245677654	riza123@gmail.com	Jl.Angkasa	Bogor
	4	Ananda	081289891111	ananda@gmail.com	Jl.Markisa No.2	Bogor
	5	Viora	081122334465	viora@gmail.com	Jl.Wijaya	Jakarta
	6	Andika	082278905678	andika@gmail.com	Jl.Wijaya	NULL

Berikut contoh pemakaian operator **OR** pada tabel Pengguna :

- **SELECT * FROM Pengguna WHERE Alamat='Jl.Wijaya' OR Kota='Jakarta';**

Hasil :

	IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
▶	1	Sarah	081289987117	sarahayuh@gmail.com	Jl.Pondok 38	Jakarta
	5	Viora	081122334465	viora@gmail.com	Jl.Wijaya	Jakarta
	6	Andika	082278905678	andika@gmail.com	Jl.Wijaya	NULL

Query di atas meminta untuk menampilkan Alamat='Jl.Wijaya' atau Kota='Jakarta'. Dapat dilihat data pengguna Sarah. Pada data tersebut *False* (Alamat = 'Jl.Pondok 38') dan *True* (Kota='Jakarta') maka dilihat dari tabel kebenaran data Sarah berupa *True*. Maka dari itu data Sarah ditampilkan. Jika melihat data NamaPengguna Viora dengan kondisi *True* (Alamat = 'Jl.Wijaya') dan *True* (Kota = 'Jakarta'), dilihat dari tabel kebenaran *True* bertemu *True* hasilnya adalah *True*. Oleh sebab itu data Viora ditampilkan. Untuk NamaPengguna Andika memiliki kondisi *True* (Alamat='Jl.Wijaya') dan *Null* (Kota=*Null*) maka dari itu dilihat dari tabel kebenaran *True* bertemu dengan *Null* hasilnya *True* data Andika ditampilkan.

- `SELECT * FROM Pengguna WHERE Alamat='Jl.Wijaya' OR Kota=NULL;`

Hasil :

	IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
▶	5	Viora	081122334465	viora@gmail.com	Jl.Wijaya	Jakarta
	6	Andika	082278905678	andika@gmail.com	Jl.Wijaya	<i>NULL</i>

Dilihat dari tabel pengguna terdapat dua baris yang memiliki Alamat = 'Jl.Wijaya', yaitu NamaPengguna Viora dan Andika. Query yang di atas meminta untuk menampilkan Alamat='Jl.Wijaya' atau Kota=*Null*. Jika melihat data NamaPengguna Viora dengan kondisi *True* (Alamat = 'Jl.Wijaya') dan *False* (Kota = 'Jakarta'), dilihat dari tabel kebenaran *True* bertemu *False* hasilnya adalah *True*. Oleh sebab itu data Viora ditampilkan. Untuk NamaPengguna Andika berarti memiliki kondisi *True* (Alamat='Jl.Wijaya') dan *Null* (Kota=*Null*) maka dari itu dilihat dari tabel kebenaran *True* bertemu dengan *Null* hasilnya *True* data Andika ditampilkan.

- `SELECT * FROM Pengguna WHERE Alamat='Jl.Wijaya' OR Kota='Bogor';`

Hasil :

	IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
▶	3	Riza	081245677654	riza123@gmail.com	Jl.Angkasa	Bogor
	4	Ananda	081289891111	ananda@gmail.com	Jl.Markisa No.2	Bogor
	5	Viora	081122334465	viora@gmail.com	Jl.Wijaya	Jakarta
	6	Andika	082278905678	andika@gmail.com	Jl.Wijaya	<i>NULL</i>

Query di atas meminta untuk menampilkan Alamat='Jl.Wijaya' atau Kota='Bogor'. Dapat dilihat data pengguna Riza. Pada data tersebut *False* (Alamat = 'Jl.Angkasa') dan *True* (Kota='Bogor') maka dilihat dari tabel kebenaran data Riza berupa *True*. Maka dari itu data Riza ditampilkan. Jika melihat data NamaPengguna Ananda dengan kondisi *False* (Alamat = 'Jl.Markisa No.2') dan *True* (Kota = 'Bogor') maka dilihat dari tabel kebenaran data Ananda berupa *True*. Maka dari itu data Ananda ditampilkan. Jika melihat data NamaPengguna Viora dengan kondisi *True* (Alamat = 'Jl.Wijaya') dan *False* (Kota = 'Jakarta'), dilihat dari tabel kebenaran *True* bertemu *False* hasilnya adalah *True*. Oleh sebab itu data Viora ditampilkan. Jika melihat data NamaPengguna Andika dengan kondisi *True* (Alamat = 'Jl.Wijaya') dan *Null* (Kota = *Null*), dilihat dari tabel kebenaran *True* bertemu *Null* hasilnya adalah *True*. Oleh sebab itu data Andika ditampilkan.

4.2 INSERT

`INSERT` merupakan perintah dalam DML untuk menambahkan sebuah data ke dalam tabel. Bentuk umum dari `INSERT` adalah:

```
INSERT INTO table_name(column_1, column_2,...) VALUES(value_1, value_2,... );
```

Penjelasan :

- `table_name` : Merupakan nama tabel dari data yang ingin diinput
- `column_1, column_2,...` : Merupakan nama kolom dari data yang ingin diinput
- `value_1, value_2,...` : Merupakan isi atau data yang akan diinput kedalam tabel

Berikut contoh dari penggunaan `INSERT` :

- ```
INSERT INTO Barang (IDBarang, NamaBarang, HargaBeli, HargaJual, Stok)
VALUES('BRG1', 'Kemeja Panjang', 50000, 60000, 10);
```

Hasil :

|   | IDBarang | NamaBarang     | HargaBeli | HargaJual | Stok |
|---|----------|----------------|-----------|-----------|------|
| ▶ | BRG1     | Kemeja Panjang | 50000     | 60000     | 10   |

Melalui contoh di atas, terdapat data yang diinput menggunakan petik satu (') dan tidak menggunakan petik satu. Hal tersebut tergantung dari tipe data yang diberikan saat pembuatan tabel. Pada tabel barang, kolom IDBarang dan NamaBarang merupakan tipe data `VARCHAR` maka perlu digunakan petik untuk penginputan data. Lalu kolom HargaBeli, HargaJual dan Stok merupakan tipe data `DECIMAL` dan `INT` sehingga tidak membutuhkan petik.

`INSERT` juga memiliki beberapa tipe, yaitu `INSERT Multiple Rows` dan `INSERT INTO SELECT`, Berikut penjelasannya :

#### 4.2.1 INSERT Multiple Rows

`INSERT multiple rows` adalah jenis penginputan yang dapat dilakukan secara bersamaan. Melalui satu query dapat dilakukan penginputan lebih dari 1 *record* atau baris dengan cara memisahkan penginputan data per-baris menggunakan koma (,). Berikut bentuk umum dari `INSERT multiple rows` :

```
INSERT INTO table_name(column_1, column_2,...) VALUES(value_1, value_2,...),
(value_1, value_2,...),...;
```

Penjelasan :

- `table_name` : Merupakan nama tabel dari data yang ingin diinput
- `column_1, column_2,...` : Merupakan nama kolom dari data yang ingin diinput
- `value_1, value_2,...` : Merupakan isi atau data yang akan diinput kedalam tabel

Contoh penginputan data menggunakan `INSERT Multiple Rows` :

- ```
INSERT INTO Barang (IDBarang, NamaBarang, HargaBeli, HargaJual, Stok)
VALUES
('BRG2', 'Celana Bahan Hitam', 60000, 65000, 20),
('BRG3', 'Rok Bahan Coklat', 60000, 65000, 50);
```

Hasil :

	IDBarang	NamaBarang	HargaBeli	HargaJual	Stok
►	BRG1	Kemeja Panjang	50000	60000	10
	BRG2	Celana Bahan Hitam	60000	65000	20
	BRG3	Rok Bahan Coklat	60000	65000	20

4.2.2 INSERT INTO SELECT

`INSERT INTO SELECT` merupakan penginput data ke dalam tabel, dimana data berasal dari hasil pernyataan `SELECT` atau berasal dari tabel lain. Berikut bentuk umum dari `INSERT INTO SELECT` :

```
INSERT INTO table_name(column_1, column_2,...) SELECT SELECT_list FROM
another_table WHERE condition;
```

Penjelasan :

- `table_name` : Merupakan nama tabel dari data yang akan diinput di dalamnya
- `column_1, column_2, ...` : Merupakan nama kolom yang akan diinput datanya
- `SELECT_list` : Merupakan nama kolom dari tabel lain yang akan diambil datanya
- `another_table` : Merupakan nama tabel yang akan diambil datanya
- `condition` : Merupakan suatu kondisi dari data yang ingin diambil

Tabel Pengguna

	IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
►	1	Sarah	081289987117	sarahayuh@gmail.com	Jl.Pondok 38	Jakarta
	2	Dyan	081267761355	dyantriandini@yahoo.com	Jl.Sari D35	Bandung
	3	Riza	081245677654	riza123@gmail.com	Jl.Angkasa	Bogor
	4	Ananda	081289891111	ananda@gmail.com	Jl.Markisa No.2	Bogor
	5	Viora	081122334465	viora@gmail.com	Jl.Wijaya	Jakarta
	6	Andika	082278905678	andika@gmail.com	Jl.Wijaya	NULL

Berikut contoh untuk `INSERT INTO SELECT`, dengan menginput tabel kosong bernama tabel Pengguna_Jakarta dan mengambil data dari tabel Pengguna :

- ```
INSERT INTO Pengguna_Jakarta (IDPengguna, NamaPengguna, NoHP, Email, Alamat, Kota)
SELECT IDPengguna, NamaPengguna, NoHP, Email, Alamat, Kota
FROM Pengguna WHERE Kota='Jakarta';
```

Hasil :

Tabel Pengguna\_Jakarta

|   | IDPengguna | NamaPengguna | NoHP         | Email               | Alamat       | Kota    |
|---|------------|--------------|--------------|---------------------|--------------|---------|
| ► | 1          | Sarah        | 081289987117 | sarahayuh@gmail.com | Jl.Pondok 38 | Jakarta |
|   | 5          | Viora        | 081122334465 | viora@gmail.com     | Jl.Wijaya    | Jakarta |

## 4.3 SELECT

Perintah `SELECT` berfungsi untuk menampilkan sebuah data. Menggunakan perintah ini dapat menampilkan data sesuai kondisi data yang kita inginkan.

### 4.3.1 SELECT Tabel

`SELECT` tabel merupakan query untuk menampilkan sebuah data. Data yang ditampilkan dapat berupa beberapa kolom atau semua data dari suatu tabel. Jika ingin menampilkan semua data pada tabel dapat menggunakan simbol bintang (\*). Berikut bentuk umum dari `SELECT` :

```
SELECT column_1, column_2,... FROM table_name;
```

Penjelasan :

- `column_1, column_2,...` : Merupakan nama kolom yang akan ditampilkan
- `table_name` : Merupakan nama tabel dari data yang ingin ditampilkan

Berikut contoh untuk menampilkan tabel barang menggunakan perintah `SELECT` :

- ```
SELECT * FROM Barang;
```

Hasil :

	IDBarang	NamaBarang	HargaBeli	HargaJual	Stock
▶	BRG0001	Kemeja Panjang	50000	60000	10
	BRG0002	Celana Bahan Hitam	60000	65000	20
	BRG0003	Rok Bahan Coklat	60000	65000	20

- ```
SELECT NamaBarang, HargaJual FROM Barang;
```

Hasil :

|   | NamaBarang         | HargaJual |
|---|--------------------|-----------|
| ▶ | Kemeja Panjang     | 60000     |
|   | Celana Bahan Hitam | 65000     |
|   | Rok Bahan Coklat   | 65000     |

### 4.3.2 SELECT Tabel Dengan Kondisi

Menampilkan sebuah tabel dapat menggunakan kondisi dengan bantuan klausa `WHERE`. Berikut bentuk umum dari `SELECT` menggunakan kondisi (`WHERE`) :

```
SELECT column_1, column_2,... FROM table_name WHERE condition;
```

Penjelasan :

- `column_1, column_2` : Merupakan nama kolom yang akan ditampilkan
- `table_name` : Merupakan nama tabel dari data yang akan ditampilkan
- `condition` : Merupakan sebuah kondisi dari data yang ingin ditampilkan

Contoh penggunaan `SELECT` menggunakan kondisi :

- ```
SELECT * FROM Pengguna WHERE Kota='Bogor';
```


Hasil :

	IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
	3	Riza	081245677654	riza123@gmail.com	Jl.Angkasa	Bogor
	4	Ananda	081289891111	ananda@gmail.com	Jl.Markisa No.2	Bogor

4.4 UPDATE

`UPDATE` adalah perintah untuk mengubah data. Menggunakan `UPDATE` kita dapat mengubah penulisan data sesuai data yang ingin kita ubah. Berikut bentuk umum dari `UPDATE` :

```
UPDATE table_name SET column_1 = expression_1, column_2= expression_2,... WHERE condition;
```

Penjelasan :

- `table_name` : Merupakan nama tabel yang datanya akan diubah
- `column_1, column_2, ...` : Merupakan nama kolom yang akan diubah
- `expression_1, expression_2, ...` : Merupakan data baru yang akan di input kedalam tabel
- `condition` : Merupakan kondisi dari data yang akan diubah

Contoh dari penggunaan `UPDATE` pada tabel pengguna :

Tabel Pengguna

	IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
▶	1	Sarah	081289987117	sarahayuh@gmail.com	Jl.Pondok 38	Jakarta
	2	Dyan	081267761355	dyantriandini@yahoo.com	Jl.Sari D35	Bandung
	3	Riza	081245677654	riza123@gmail.com	Jl.Angkasa	Bogor
	4	Ananda	081289891111	ananda@gmail.com	Jl.Markisa No.2	Bogor
	5	Viora	081122334465	viora@gmail.com	Jl.Wijaya	Jakarta
	6	Andika	082278905678	andika@gmail.com	Jl.Wijaya	NULL

- Mengubah Alamat dan Kota pada data IDPengguna = 5 atas nama Viora.

```
UPDATE Pengguna
SET Alamat = 'Jl.Jati Padang', Kota = 'Depok'
WHERE IDPengguna=5;
```

Hasil :

	IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
▶	1	Sarah	081289987117	sarahayuh@gmail.com	Jl.Pondok 38	Jakarta
	2	Dyan	081267761355	dyantriandini@yahoo.com	Jl.Sari D35	Bandung
	3	Riza	081245677654	riza123@gmail.com	Jl.Angkasa	Bogor
	4	Ananda	081289891111	ananda@gmail.com	Jl.Markisa No.2	Bogor
	5	Viora	081122334465	viora@gmail.com	Jl.Jati Padang	Depok
	6	Andika	082278905678	andika@gmail.com	Jl.Wijaya	NULL

4.5 DELETE

`DELETE` merupakan perintah untuk menghapus data. `DELETE` tidak dapat menghapus table ataupun database. `DELETE` hanya digunakan untuk menghapus baris atau *record*. Berikut bentuk umum yang digunakan untuk `DELETE` :

```
DELET FROM table_name WHERE condition;
```

Penjelasan :

- `table_name` : Merupakan nama tabel dari data yang akan dihapus
- `condition` : Merupakan kondisi dari data yang akan dihapus

Tabel Pengguna

	IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
▶	1	Sarah	081289987117	sarahayuh@gmail.com	Jl.Pondok 38	Jakarta
	2	Dyan	081267761355	dyantriandini@yahoo.com	Jl.Sari D35	Bandung
	3	Riza	081245677654	riza123@gmail.com	Jl.Angkasa	Bogor
	4	Ananda	081289891111	ananda@gmail.com	Jl.Markisa No.2	Bogor
	5	Viora	081122334465	viora@gmail.com	Jl.Wijaya	Jakarta
	6	Andika	082278905678	andika@gmail.com	Jl.Wijaya	NULL

Contoh penggunaan perintah `DELETE` pada tabel Pengguna :

- Menghapus baris data IDPengguna = 6 atas nama Andika

```
DELETE FROM Pengguna WHERE IDPengguna = 6;
```

Hasil :

	IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
▶	1	Sarah	081289987117	sarahayuh@gmail.com	Jl.Pondok 38	Jakarta
	2	Dyan	081267761355	dyantriandini@yahoo.com	Jl.Sari D35	Bandung
	3	Riza	081245677654	riza123@gmail.com	Jl.Angkasa	Bogor
	4	Ananda	081289891111	ananda@gmail.com	Jl.Markisa No.2	Bogor
	5	Viora	081122334465	viora@gmail.com	Jl.Jati Padang	Depok

REFERENSI

- [1] Mysqлтutorial. "MySQL AND Operator", <https://www.mysqлтutorial.org/mysql-and/>, diakses 23 Juli 2020.
- [2] Mysqлтutorial. "MySQL OR Operator", <https://www.mysqлтutorial.org/mysql-or/>, diakses 23 Juli 2020.
- [3] Mysqлтutorial. "MySQL Insert Multiple Rows", <https://www.mysqлтutorial.org/mysql-insert-multiple-rows/>, diakses 23 Juli 2020.
- [4] Mysqлтutorial. "MySQL INSERT INTO SELECT", <https://www.mysqлтutorial.org/mysql-insert-into-select/>, diakses 23 Juli 2020.
- [5] Mysqлтutorial. "MySQL SELECT", <https://www.mysqлтutorial.org/mysql-select-statement-query-data.aspx>, diakses 23 Juli 2020.
- [6] Mysqлтutorial. "MySQL UPDATE", <https://www.mysqлтutorial.org/mysql-update-data.aspx>, diakses 23 Juli 2020.
- [7] Mysqлтutorial. "MySQL DELETE", <https://www.mysqлтutorial.org/mysql-delete-statement.aspx>, diakses 23 Juli 2020.