

**LAPORAN PRAKTIKUM  
BASIS DATA**

**MODUL IV  
MANIPULASI DAN RETRIEVE DATA (BAGIAN 2)**



Disusun Oleh:

Septiandi Nugraha

21104060

SE05-B

**PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK  
FAKULTAS INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

## I. Tujuan

1. Dapat menggunakan select statement untuk mengambil data dari satu table
2. Dapat menyeleksi data berdasarkan kondisi tertentu

## II. Dasar Teori

Select statement digunakan sebagai operasi dalam basis data untuk mengambil sejumlah baris data yang memenuhi predikat yang diberikan. Predikat ini mengacu pada kondisi yang akan dipenuhi dalam operasi seleksi. Berikut perintah-perintah SELECT :

- a. Memberikan nama lain pada kolom :

```
SELECT namakolomlama AS namakolombaru FROM namatabel;
```

Contoh :

```
SELECT jenis AS judul_buku FROM master_buku;
```

- b. Menggunakan alias untuk nama tabel :

```
SELECT namaalias.namakolom1, namaalias.namakolom2 FROM  
namatabel namaalias;
```

Contoh :

```
SELECT B.judul_buku, B.pengarang FROM master_buku B;
```

- c. Menampilkan data lebih dari dua tabel :

```
SELECT * FROM namatabel1, namatabel2, namatabel-n;
```

Contoh :

```
SELECT * FROM master_buku;
```

- d. Nested Queries / Subquery (IN, NOT IN, EXISTS, NOT EXISTS)

Subquery yaitu query yang terdiri dari beberapa query. Dengan menggunakan subquery, hasil dari query akan menjadi bagian dari query di atasnya. Subquery terletak di dalam klausa WHERE atau HAVING. Pada klausa WHERE, subquery digunakan untuk memilih baris-baris tertentu yang kemudian digunakan oleh query. Sedangkan pada klausa HAVING, subquery digunakan untuk memilih kelompok baris yang kemudian digunakan oleh query.

Contoh penggunaan **IN** :

```
SELECT * FROM costumers WHERE country IN ('Germany',  
'France', 'UK');
```

Contoh penggunaan **NOT IN** :

```
SELECT * FROM costumers WHERE country NOT IN ('Germany',  
'France', 'UK');
```

Contoh penggunaan **EXIST** :

```
SELECT SupplierName  
FROM Suppliers  
WHERE EXISTS (SELECT ProductName FROM Products WHERE  
SupplierId = Suppliers.supplierId AND Price = 22);
```

Contoh penggunaan **NOT EXIST** :

```
SELECT SupplierName  
FROM Suppliers  
WHERE NOT EXISTS (SELECT ProductName FROM Products WHERE  
SupplierId = Suppliers.supplierId AND Price = 22);
```

Pada contoh di atas : **SELECT ProductName FROM Products** disebut subquery, sedangkan: **SELECT SupplierName FROM Suppliers** berkedudukan sebagai query.

e. Operator comparison ANY dan ALL

Operator ANY digunakan berkaitan dengan subquery. Operator ini menghasilkan TRUE (benar) jika paling tidak salah satu perbandingan dengan hasil subquery menghasilkan nilai TRUE.

Contoh penggunaan ANY :

```
SELECT ProductName, Price  
FROM Products  
WHERE ProductID = ANY (SELECT ProductID FROM OrderDetails  
WHERE Quantity = 10);
```

Operator ALL digunakan untuk melakukan perbandingan dengan subquery. Kondisi dengan ALL menghasilkan nilai TRUE (benar) jika subquery tidak menghasilkan apapun atau jika perbandingan menghasilkan TRUE untuk setiap nilai query terhadap hasil subquery.

Contoh penggunaan ALL :

```
SELECT ProductName  
FROM Products  
WHERE ProductID = ALL (SELECT ProductID FROM OrderDetails  
WHERE Quantity = 10);
```

f. Penggunaan ORDER BY

Klausula ORDER BY digunakan untuk mengurutkan data berdasarkan kolom tertentu sesuai dengan tipe data yang dimiliki.

Contoh penggunaan ORDER BY :

```
SELECT * FROM Customers  
ORDER BY Country;
```

atau tambahkan ASC untuk pengurutan secara ascending (menaik) :

```
SELECT * FROM Customers  
ORDER BY Country ASC;
```

atau tambahkan DESC untuk pengurutan secara descending (menurun) :

```
SELECT * FROM Customers  
ORDER BY Country DESC;
```

g. DISTINCT

Distinct adalah kata kunci ini untuk menghilangkan duplikasi. Sebagai Contoh: Sebuah tabel pelanggan yang berisi nama dan kota asal dengan beberapa record isi dan beberapa kota asal yang sama. Kemudian ketikkan perintah berikut :

```
SELECT DISTINCT Country FROM Customers;
```

Dengan perintah di atas maka nama kota yang sama hanya akan ditampilkan satu saja.

h. UNION, INTERSECT dan EXCEPT

UNION merupakan operator yang digunakan untuk menggabungkan hasil query, dengan ketentuan jumlah, nama dan tipe kolom dari masing-masing tabel yang akan ditampilkan datanya harus sama. Berikut ini perintah untuk memperoleh data pada tabel film dimana jenisnya action dan horor :

```
SELECT City, Country FROM Customers  
WHERE Country='Germany'
```

**UNION**

```
SELECT City, Country FROM Customers  
WHERE Country='France';
```

Perintah di atas identik dengan :

```
SELECT City, Country FROM Customers  
WHERE Country='Germany' OR Country='France';
```

Namun tidak semua penggabungan dapat dilakukan dengan OR, yaitu jika bekerja pada dua tabel atau lebih.

INTERSECT merupakan operator yang digunakan untuk memperoleh data dari dua buah query dimana data yang ditampilkan adalah yang memenuhi kedua query tersebut dengan ketentuan jumlah, nama dan tipe kolom dari masing-masing tabel yang akan ditampilkan datanya harus sama.

```
SELECT * FROM namatabel1 INTERSECT SELECT * FROM  
namatabel2 ;
```

Pada MySQL tidak terdapat operator INTERSECT namun sebagai gantinya dapat menggunakan operator IN seperti pada bagian (d) Nested Queries. EXCEPT / Set Difference merupakan operator yang digunakan untuk memperoleh data dari dua buah query dimana data yang ditampilkan adalah data yang ada pada hasil query 1 dan tidak terdapat pada data dari hasil query 2 dengan ketentuan jumlah, nama dan tipe kolom dari masing-masing tabel yang akan ditampilkan datanya harus sama.

```
SELECT * FROM namatabel1 EXCEPT SELECT * FROM  
namatabel2;
```

Pada MySQL tidak terdapat operator EXCEPT namun sebagai gantinya dapat menggunakan operator NOT IN seperti contoh (d) pada bagian Nested Queries.

### III. Praktikum

1. Buatlah tabel dosen sebagai berikut

Field	Tipe Data	Ukuran
id_dosen	Varchar	6
nama_dosen	Varchar	20



Data DESC dosen

	Field	Type	Null	Key	Default	Extra
	varchar	blob	varchar	varchar	text	varchar
1	id_dosen	varchar(6)	NO	PRI	(NULL)	
2	nama_dosen	varchar(20)	YES		(NULL)	
3	jk	char(1)	YES		(NULL)	
4	no_hp	varchar(12)	YES		(NULL)	
5	gaji	int(11)	YES		(NULL)	
6	tunjangan_jabatan	int(11)	YES		(NULL)	

Ln 15, Col 12 Spaces: 4 UTF-8 CRLF SQL Go Live Spell

## 2. Isi data tabel dengan data-data berikut!

id_dosen	nama_dosen	jk	no_hp	gaji	tunjangan_jabatan	id_kaprodi	id_prodi
D101	Renaldi	L	0812344	1700000	500000	K301	P021
D102	Abdul	L	0812365	1950000	450000	K302	P022
D103	Viona	P	0815644	2000000	560000	K304	P034
D104	Wita	P	0815663	2500000	760000	K301	P022
D105	Atika	P	0812563	1900000	860000	K302	P034
D106	Zehan	L	0812983	1200000	0	K304	P021
D107	Sari	P	0812233	1550000	0	K301	P022

```
INSERT INTO dosen
VALUES ('D101', 'Renaldi', 'L', '0812344', 1700000, 500000, 'K301', 'P021'),
('D102', 'Abdul', 'L', '0812365', 1950000, 450000, 'K302', 'P022'),
('D103', 'Viona', 'P', '0815644', 2000000, 560000, 'K304', 'P034'),
('D104', 'Wita', 'P', '0815663', 2500000, 760000, 'K301', 'P022'),
('D105', 'Atika', 'P', '0812563', 1900000, 860000, 'K302', 'P034'),
('D106', 'Zehan', 'L', '0812983', 1200000, 0, 'K304', 'P021'),
('D107', 'Sari', 'P', '0812233', 1550000, 0, 'K301', 'P022');
```

INSERT INTO dosen

```
VALUES ('D101', 'Renaldi', 'L', '0812344', 1700000, 500000, 'K301', 'P021'),
('D102', 'Abdul', 'L', '0812365', 1950000, 450000, 'K302', 'P022'),
('D103', 'Viona', 'P', '0815644', 2000000, 560000, 'K304', 'P034'),
('D104', 'Wita', 'P', '0815663', 2500000, 760000, 'K301', 'P022'),
('D105', 'Atika', 'P', '0812563', 1900000, 860000, 'K302', 'P034'),
('D106', 'Zehan', 'L', '0812983', 1200000, 0, 'K304', 'P021'),
('D107', 'Sari', 'P', '0812233', 1550000, 0, 'K301', 'P022');
```

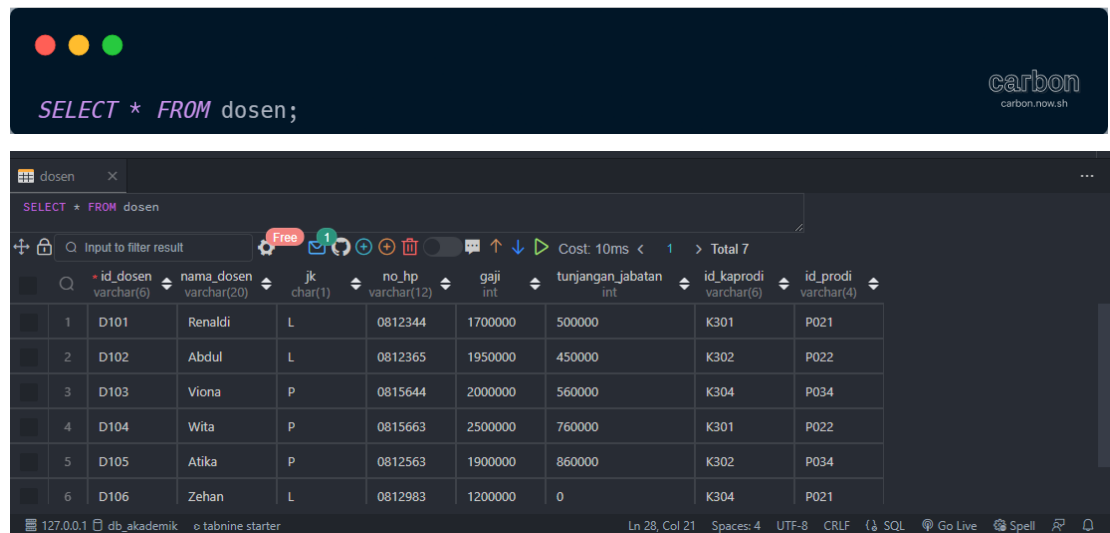
Cost: 581ms

INSERT INTO dosen VALUES ('D101', 'Renaldi', 'L', '0812344', 1700000, 500000, 'K301', 'P021'), ('D102', 'Abdul', 'L', '0812365', 1950000, 450000, 'K302', 'P022'), ('D103', 'Viona', 'P', '0815644', 2000000, 560000, 'K304', 'P034'), ('D104', 'Wita', 'P', '0815663', 2500000, 760000, 'K301', 'P022'), ('D105', 'Atika', 'P', '0812563', 1900000, 860000, 'K302', 'P034'), ('D106', 'Zehan', 'L', '0812983', 1200000, 0, 'K304', 'P021'), ('D107', 'Sari', 'P', '0812233', 1550000, 0, 'K301', 'P022')

AffectedRows : 7

Ln 25, Col 73 Spaces: 4 UTF-8 CRLF SQL Go Live Spell

3. Tampilkan semua kolom tabel!

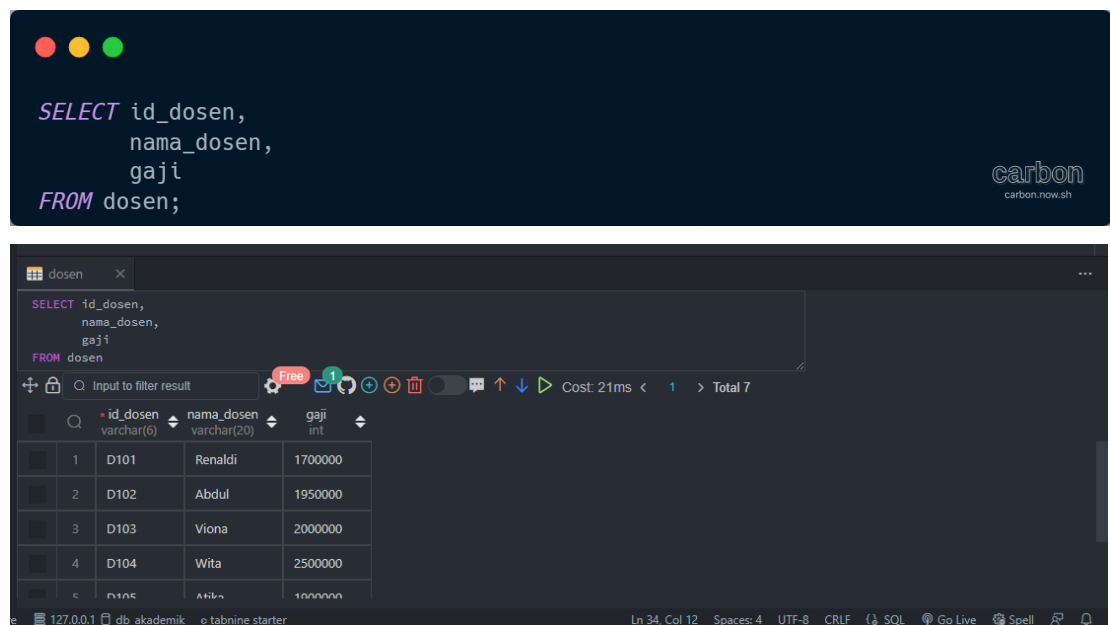


Carbon CLI terminal showing a SQL query and its result in a table.

```
SELECT * FROM dosen;
```

	id_dosen varchar(6)	nama_dosen varchar(20)	jk char(1)	no_hp varchar(12)	gaji int	tunjangan_jabatan int	id_kaprodi varchar(6)	id_prodi varchar(4)
1	D101	Renaldi	L	0812344	1700000	500000	K301	P021
2	D102	Abdul	L	0812365	1950000	450000	K302	P022
3	D103	Viona	P	0815644	2000000	560000	K304	P034
4	D104	Wita	P	0815663	2500000	760000	K301	P022
5	D105	Atika	P	0812563	1900000	860000	K302	P034
6	D106	Zehan	L	0812983	1200000	0	K304	P021

4. Tampilkan kolom id\_dosen, nama\_dosen dan gaji saja!



Carbon CLI terminal showing a SQL query and its result in a table.

```
SELECT id_dosen,  
       nama_dosen,  
       gaji  
FROM dosen;
```

	id_dosen varchar(6)	nama_dosen varchar(20)	gaji int
1	D101	Renaldi	1700000
2	D102	Abdul	1950000
3	D103	Viona	2000000
4	D104	Wita	2500000
5	D105	Atika	1900000
6	D106	Zehan	1200000

5. Tampilkan kolom id\_dosen, nama\_dosen, gaji, tunjangan\_jabatan dan sebuah kolom baru yaitu tunjangan\_jabatan + gaji yang berisi jumlah tunjangan\_jabatan dan gaji !



```

SELECT id_dosen,
       nama_dosen,
       gaji,
       tunjangan_jabatan,
       gaji + tunjangan_jabatan AS 'Gaji Tunjangan'
FROM dosen;

```

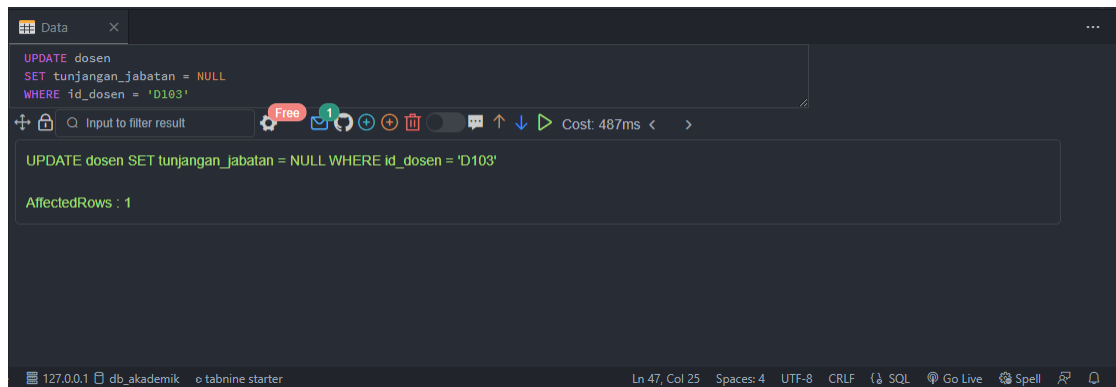
6.

Ubah tunjanganJabatan menjadi NULL untuk dosen dengan idDosen = D103.  
Kemudia lakukan kembali percobaan 5.

```

UPDATE dosen
SET tunjangan_jabatan = NULL
WHERE id_dosen = 'D103';

```

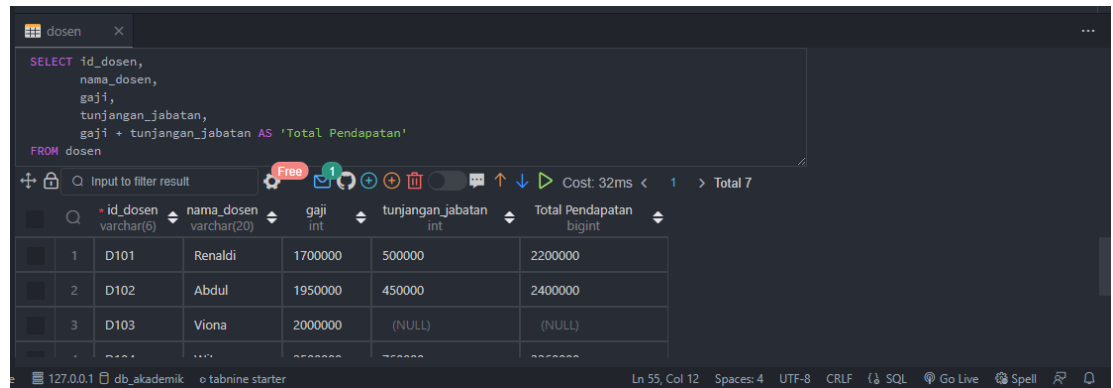


7. Seperti percobaan 5, tampilkan kolom idDosen, NamaDosen, gaji, tunjanganJabatan dan sebuah kolom baru (gunakan alias) yaitu total\_pendapatan yang berisi jumlah tunjanganJabatan dan gaji !

```

SELECT id_dosen,
       nama_dosen,
       gaji,
       tunjangan_jabatan,
       gaji + tunjangan_jabatan AS 'Total Pendapatan'
FROM dosen;

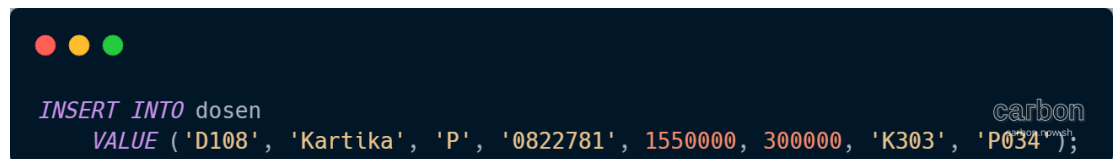
```



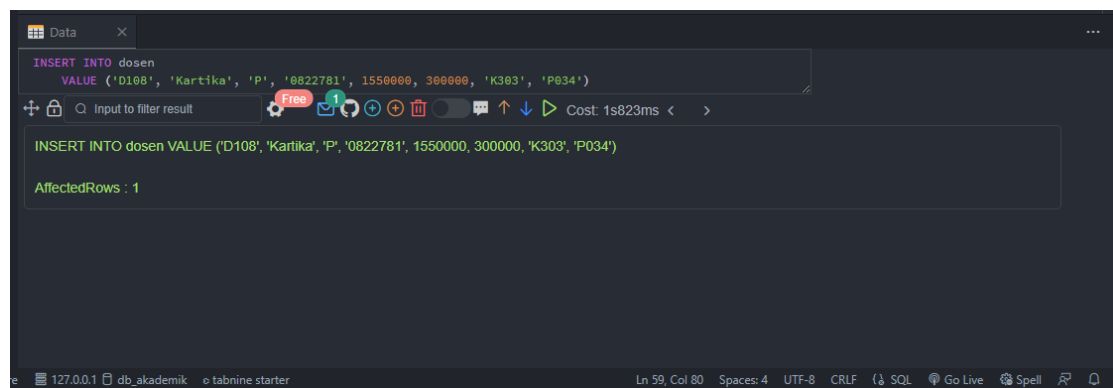
```
SELECT id_dosen,
nama_dosen,
gaji,
tunjangan_jabatan,
gaji + tunjangan_jabatan AS 'Total Pendapatan'
FROM dosen
```

	id_dosen varchar(6)	nama_dosen varchar(20)	gaji int	tunjangan_jabatan int	Total Pendapatan bigint
1	D101	Renaldi	1700000	500000	2200000
2	D102	Abdul	1950000	450000	2400000
3	D103	Viona	2000000	(NULL)	(NULL)

8. Tambahkan record baru dengan value: D108, Kartika, P, 0822781, 1550000, 300000, K303, P034 !



```
INSERT INTO dosen
VALUE ('D108', 'Kartika', 'P', '0822781', 1550000, 300000, 'K303', 'P034');
```



```
INSERT INTO dosen
VALUE ('D108', 'Kartika', 'P', '0822781', 1550000, 300000, 'K303', 'P034')
```

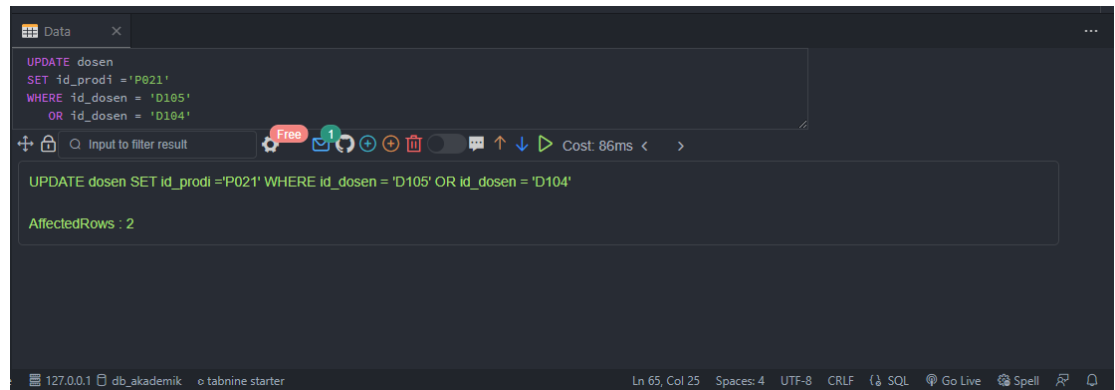
```
INSERT INTO dosen VALUE ('D108', 'Kartika', 'P', '0822781', 1550000, 300000, 'K303', 'P034')
```

AffectedRows : 1

9. Untuk dosen yang ber-id D104 dan D105 ubah idProdi menjadi P021!

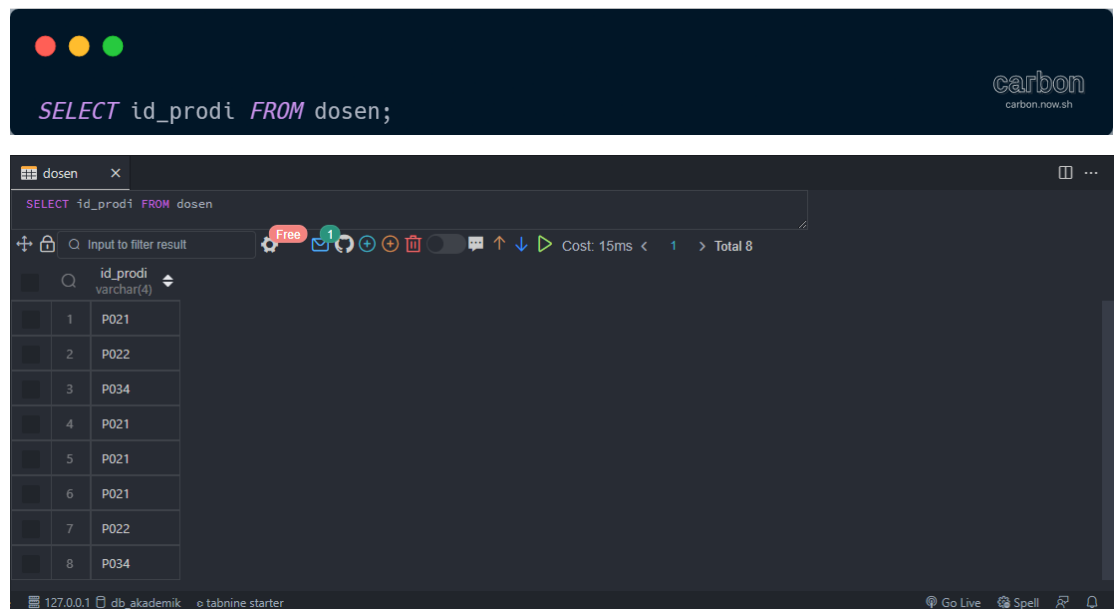


```
UPDATE dosen
SET id_prodi ='P021'
WHERE id_dosen = 'D105'
OR id_dosen = 'D104';
```



The screenshot shows a SQL editor window titled 'Data'. The SQL statement entered is: `UPDATE dosen SET id_prodi = 'P021' WHERE id_dosen = 'D105' OR id_dosen = 'D104'`. Below the statement, the execution results are displayed: `UPDATE dosen SET id_prodi = 'P021' WHERE id_dosen = 'D105' OR id_dosen = 'D104'` and `AffectedRows : 2`. The status bar at the bottom indicates the connection is to `127.0.0.1 db_akademik` using the `tabnine starter` driver.

10. Sekarang tampilkan kolom idProdi saja !



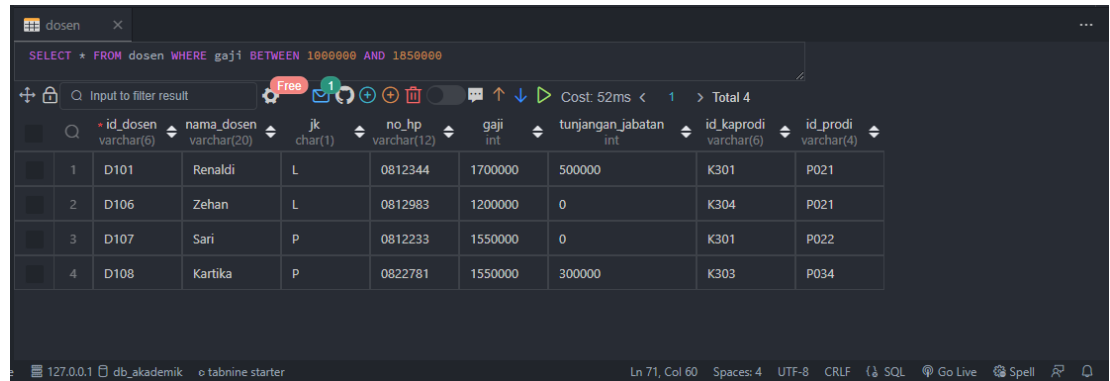
The screenshot shows a SQL editor window titled 'dosen'. The SQL statement entered is: `SELECT id_prodi FROM dosen;`. Below the statement, the execution results are displayed as a table with 8 rows. The status bar at the bottom indicates the connection is to `127.0.0.1 db_akademik` using the `tabnine starter` driver.

	id_prodi
1	P021
2	P022
3	P034
4	P021
5	P021
6	P021
7	P022
8	P034

11. Tampilkan pegawai yang gajinya antara 1000000 - 1850000 !

```
SELECT * FROM dosen WHERE gaji BETWEEN 1000000 AND 1850000;
```

carbon  
carbon.now.sh



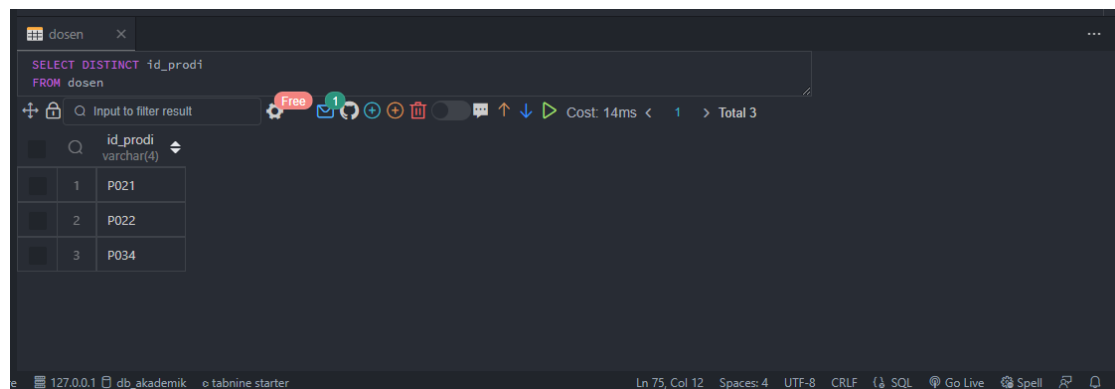
The screenshot shows a SQL query editor with the query: `SELECT * FROM dosen WHERE gaji BETWEEN 1000000 AND 1850000`. The results are displayed in a table with 10 columns: `id_dosen`, `nama_dosen`, `jk`, `no_hp`, `gaji`, `tunjangan_jabatan`, `id_kaprodi`, and `id_prodi`. The table contains 4 rows of data.

	id_dosen	nama_dosen	jk	no_hp	gaji	tunjangan_jabatan	id_kaprodi	id_prodi
1	D101	Renaldi	L	0812344	1700000	500000	K301	P021
2	D106	Zehan	L	0812983	1200000	0	K304	P021
3	D107	Sari	P	0812233	1550000	0	K301	P022
4	D108	Kartika	P	0822781	1550000	300000	K303	P034

12. Tampilkan id\_prodi dengan menghilangkan data id\_prodi yang duplikasi !

```
SELECT DISTINCT id_prodi  
FROM dosen;
```

carbon  
carbon.now.sh



The screenshot shows a SQL query editor with the query: `SELECT DISTINCT id_prodi FROM dosen`. The results are displayed in a table with 2 columns: `id_prodi`. The table contains 3 rows of data.

	id_prodi
1	P021
2	P022
3	P034

13. Tampilkan dosen yang jenis kelamin laki-laki !

```
SELECT nama_dosen,  
       jk  
FROM dosen  
WHERE jk = 'L';
```

carbon  
carbon.now.sh

dosen

```
SELECT nama_dosen,
       jk
FROM dosen
WHERE jk = 'L'
```

Cost: 20ms < 1 > Total 3

	nama_dosen varchar(20)	jk char(1)
1	Renaldi	L
2	Abdul	L
3	Zehan	L

127.0.0.1 db\_akademik tabnine starter Ln 81, Col 16 Spaces: 4 UTF-8 CRLF SQL Go Live Spell

14. Tampilkan seluruh dosen menurut abjad dari Z-A !

```
SELECT *
FROM dosen
ORDER BY nama_dosen DESC;
```

carbon  
carbon.now.sh

dosen

```
SELECT *
FROM dosen
ORDER BY nama_dosen DESC
```

Cost: 51ms < 1 > Total 8

	id_dosen varchar(6)	nama_dosen varchar(20)	jk char(1)	no_hp varchar(12)	gaji int	tunjangan_jabatan int	id_kaprodi varchar(6)	id_prodi varchar(4)
1	D106	Zehan	L	0812983	1200000	0	K304	P021
2	D104	Wita	P	0815663	2500000	760000	K301	P021
3	D103	Viona	P	0815644	2000000	(NULL)	K304	P034
4	D107	Sari	P	0812233	1550000	0	K301	P022
5	D101	Renaldi	L	0812344	1700000	500000	K301	P021

127.0.0.1 db\_akademik tabnine starter Ln 86, Col 26 Spaces: 4 UTF-8 CRLF SQL Go Live Spell

15. Tampilkan seluruh dosen menurut abjad dari A-Z !

```
SELECT *
FROM dosen
ORDER BY nama_dosen ASC;
```

carbon  
carbon.now.sh

</

#### IV. Kesimpulan

Dalam basis data, select statement digunakan sebagai operasi untuk mengambil sejumlah baris data yang memenuhi predikat yang diberikan. Select statement terdiri dari beberapa bagian untuk mengatur dan mengambil data dari suatu tabel, serta dapat menyeleksi data berdasarkan kondisi tertentu.