

SELECT LANJUTAN

AND

Struktur Query

```
select kolom1,kolom2 from nama_tabel, where kolom1="nilai1" AND kolom2="nilai2;
```

Contoh Query

```
select warna,pemilik from mobil where warna='HITAM' AND pemilik='RANI';
```

Hasil

```
MariaDB [rental_adel]> select warna,pemilik from mobil where warna='HITAM' AND pemilik='RANI';
+-----+-----+
| warna | pemilik |
+-----+-----+
| HITAM | RANI    |
| HITAM | RANI    |
+-----+-----+
2 rows in set (0.078 sec)
```

Analisis

1. `SELECT warna, pemilik`: Ini menentukan bahwa kita ingin memilih kolom warna dan pemilik dari tabel mobil.
2. `FROM mobil`: Ini menentukan tabel yang kolomnya ingin kita pilih, yaitu tabel mobil.
3. `WHERE warna='HITAM' AND pemilik='RANI'`: Ini menentukan kondisi yang harus dipenuhi suatu baris agar dapat dipilih. Dalam hal ini kolom warna harus 'HITAM' dan kolom pemilik harus 'RANI'. Kata kunci AND menetapkan bahwa kedua kondisi harus benar agar suatu baris dapat dipilih.

Kesimpulan

Query di atas mempermudah kita dalam mencari data yang akurat tanpa harus mengecek semua..

OR

Struktur Query

```
select kolom1,kolom2 from nama_tabel, where kolom1="nilai1" AND kolom2="nilai2;
```

Contoh Query

```
select warna,pemilik from mobil where warna='HITAM' AND pemilik='RANI';
```

Hasil

```
MariaDB [rental_adel]> select warna,pemilik from mobil where warna='HITAM' OR pemilik='RANI';
+-----+-----+
| warna | pemilik |
+-----+-----+
| SILVER | RANI    |
| HITAM  | RANI    |
| HITAM  | RANI    |
+-----+-----+
3 rows in set (0.001 sec)
```

Analisis

1. **SELECT warna, pemilik**: Bagian query ini menentukan kolom yang ingin saya ambil dari tabel mobil. Dalam hal ini, Anda ingin mengambil kolom warna (warna) dan pemilik (pemilik).
2. **FROM mobil**: Bagian query ini menentukan tabel tempat saya ingin mengambil data. Dalam hal ini, Anda ingin mengambil data dari tabel mobil.
3. **WHERE warna='HITAM' AND pemilik='RANI'**: Bagian query ini menentukan kondisi yang harus dipenuhi data agar dapat diambil. Dalam hal ini, saya ingin mengambil data dimana warna (warna) adalah 'HITAM' (hitam) dan pemilik (pemilik) adalah 'RANI'.

Kesimpulan

perintah tersebut digunakan untuk mencari mobil dengan warna 'HITAM' yang dimiliki oleh 'RANI' dari tabel "mobil".

BETWEEN - AND

Struktur Query

```
select * from nama_tabel where kolom1 between nilai1 AND nilai2;
```

Contoh Query

```
select * from mobil where harga_rental between 100000 AND 150000;
```

Hasil

```
MariaDB [rental_adel]> select * from mobil where harga_rental between 100000 AND 150000;
```

id_mobil	no_plat	no_mesin	warna	pemilik	peminjam	harga_rental
2	DD 2440 AX	BCS1120	MERAH	ALYA	IRA	100000
4	DD 2210	UQL1029	HITAM	RANI	NULL	150000
5	DD 1111 AD	CJH1011	HITAM	RANI	NULL	100000

```
3 rows in set (0.001 sec)
```

Analisis

SELECT *: Bagian query ini menentukan bahwa Anda ingin mengambil semua kolom dari tabel seluler.

FROM mobil: Bagian query ini menentukan tabel tempat Anda ingin mengambil data. Dalam hal ini, Anda ingin mengambil data dari tabel mobile.

WHERE harga_rental BETWEEN 100000 AND 150000: Bagian query ini menentukan kondisi yang harus dipenuhi data agar dapat diambil. Dalam hal ini, Anda ingin mengambil data yang **harga_rental** (harga sewa) antara 100.000 dan 150.000.

Kesimpulan

Perintah tersebut digunakan untuk mencari mobil dengan harga sewa antara 100.000 hingga 150.000.

NOT BETWEEN

Struktur Query

```
select * from nama_tabel where kolom1 not between nilai1 AND nilai2;
```

Contoh Query

```
select * from mobil where harga_rental not between 1000000 AND 20000;
```

Hasil

```
MariaDB [rental_adel]> select * from mobil where harga_rental not between 1000000 AND 20000;
```

id_mobil	no_plat	no_mesin	warna	pemilik	peminjam	harga_rental
1	DD 2650 XY	ACX3560	HITAM	ADEL	ILA	50000
2	DD 2440 AX	BCS1120	MERAH	ALYA	IRA	100000
3	B 1617 QC	LSQ1112	SILVER	RANI	STEAL	50000
4	DD 2210	UQL1029	HITAM	RANI	NULL	150000
5	DD 1111 AD	CJH1011	HITAM	RANI	NULL	100000

```
5 rows in set (0.001 sec)
```

Analisis

1. `SELECT *` : Bagian Query ini menentukan bahwa Anda ingin mengambil semua kolom dari tabel seluler.
2. `FROM mobil` : Bagian query ini menentukan tabel tempat Anda ingin mengambil data. Dalam hal ini, Anda ingin mengambil data dari tabel mobil.
3. `WHERE harga_rental not between 1000000 and 20000` : Bagian query ini menentukan kondisi yang harus dipenuhi data agar dapat diambil. Dalam hal ini, Anda ingin mengambil data yang harga_rental (harga sewa) bukan antara 1.000.000 dan 20.000.

Kesimpulan

Perintah tersebut digunakan untuk mencari mobil dengan harga sewa di luar rentang 20.000 hingga 1.000.000.

<=

Struktur Query

```
select * from nama_tabel where kolom1 <= nilai1;
```

Contoh Query

```
select * from mobil where harga_rental <= 50000;
```

Hasil

```
MariaDB [rental_adel]> select * from mobil where harga_rental <= 50000;
+-----+-----+-----+-----+-----+-----+-----+
| id_mobil | no_plat   | no_mesin | warna  | pemilik | peminjam | harga_rental |
+-----+-----+-----+-----+-----+-----+-----+
| 1        | DD 2650 XY | ACX3560  | HITAN  | ADEL    | ILA      | 50000         |
| 3        | B 1617 QC  | LSQ1112  | SILVER | RANI    | STEA     | 50000         |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.002 sec)
```

Analisis

- 1. `SELECT *` : Bagian kueri ini menentukan bahwa Anda ingin mengambil semua kolom dari tabel seluler.
- 2. `FROM mobil` : Bagian kueri ini menentukan tabel tempat Anda ingin mengambil data. Dalam hal ini, Anda ingin mengambil data dari tabel mobile.
- 3. `WHERE harga_rental <= 50000` : Bagian kueri ini menentukan kondisi yang harus dipenuhi data agar dapat diambil. Dalam hal ini, Anda ingin mengambil data dimana harga_rental (harga sewa) kurang dari atau sama dengan 50.000.

Kesimpulan

Perintah tersebut digunakan untuk mencari mobil dengan harga sewa yang kurang dari atau sama dengan 50.000.

>=

Struktur Query

```
select * from nama_tabel where kolom1 <= nilai1;
```

Contoh Query

```
select * from mobil where harga_rental >= 50000;
```

Hasil

```
MariaDB [rental_adel]> select * from mobil where harga_rental >= 50000;
```

id_mobil	no_plat	no_mesin	warna	pemilik	peminjam	harga_rental
1	DD 2650 XY	ACX3560	HITAM	ADEL	ILA	50000
2	DD 2440 AX	BCS1120	MERAH	ALYA	IRA	100000
3	B 1617 QC	LSQ1112	SILVER	RANI	STEAL	50000
4	DD 2210	UQL1029	HITAM	RANI	NULL	150000
5	DD 1111 AD	CJH1011	HITAM	RANI	NULL	100000

```
5 rows in set (0.001 sec)
```

Analisis

1. **SELECT *** : Bagian kueri ini menentukan bahwa Anda ingin mengambil semua kolom dari tabel seluler.
2. **FROM mobil** : Bagian kueri ini menentukan tabel tempat Anda ingin mengambil data. Dalam hal ini, Anda ingin mengambil data dari tabel mobil.
3. **WHERE harga_rental >= 50000** : Bagian query ini menentukan kondisi yang harus dipenuhi data agar dapat diambil. Dalam hal ini, Anda ingin mengambil data dimana harga_rental (harga sewa) lebih besar atau sama dengan 50.000.

Kesimpulan

Hasil dari perintah tersebut akan mengembalikan semua baris dari tabel "mobil" di mana nilai kolom "harga_rental" lebih besar dari atau sama dengan 50.000. Semua kolom dari baris-baris yang memenuhi kondisi tersebut akan ditampilkan dalam hasilnya.

<> atau !=

Struktur Query

```
select * from nama_tabel where kolom1 <> nilai1;
```

Contoh Query

```
select * from mobil where harga_rental <> 50000;
```

Hasil

```
MariaDB [rental_adel]> select * from mobil where harga_rental <> 50000;
+-----+-----+-----+-----+-----+-----+-----+
| id_mobil | no_plat | no_mesin | warna | pemilik | peminjam | harga_rental |
+-----+-----+-----+-----+-----+-----+-----+
| 2 | DD 2440 AX | BCS1120 | MERAH | ALYA | IRA | 100000 |
| 4 | DD 2210 | UQL1029 | HITAM | RANI | NULL | 150000 |
| 5 | DD 1111 AD | CJH1011 | HITAM | RANI | NULL | 100000 |
+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.002 sec)
```

Analisis

- 1. `SELECT *` : Bagian kueri ini menentukan bahwa Anda ingin mengambil semua kolom dari tabel seluler.
- 2. `FROM mobil` : Bagian kueri ini menentukan tabel tempat Anda ingin mengambil data. Dalam hal ini, Anda ingin mengambil data dari tabel mobile.
- 3. `WHERE harga_rental <> 50000` : Bagian kueri ini menentukan kondisi yang harus dipenuhi data agar dapat diambil. Dalam hal ini, Anda ingin mengambil data yang harga_rental (harga sewa) tidak sama dengan 50.000.

Kesimpulan

Hasil dari perintah tersebut akan mengembalikan semua baris dari tabel "mobil" di mana nilai kolom "harga_rental" tidak sama dengan 50.000. Semua kolom dari baris-baris yang tidak memenuhi kondisi tersebut akan ditampilkan dalam hasilnya.

IN

Stuktur Query

```
select * from nama_tabel where kolom in('nilai1','nilai2');
```

Contoh Query

```
select * from mobil where warna in('silver','merah');
```

Hasil

```
MariaDB [rental_adel]> select * from mobil where warna in('silver','merah');
+-----+-----+-----+-----+-----+-----+-----+
| id_mobil | no_plat   | no_mesin | warna  | pemilik | peminjam | harga_rental |
+-----+-----+-----+-----+-----+-----+-----+
|         2 | DD 2440 AX | BCS1120  | MERAH | ALYA    | IRA      | 100000        |
|         3 | B 1617 QC  | LSQ1112  | SILVER | RANI    | STEA     | 50000         |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.002 sec)
```

Analisis

1. `SELECT *` : Bagian kueri ini menentukan bahwa Anda ingin mengambil semua kolom dari tabel seluler.
2. `FROM mobil` : Bagian kueri ini menentukan tabel tempat Anda ingin mengambil data. Dalam hal ini, Anda ingin mengambil data dari tabel mobile.
3. `WHERE warna IN ('silver', 'merah')` : Bagian kueri ini menentukan kondisi yang harus dipenuhi data agar dapat diambil. Dalam hal ini, Anda ingin mengambil data yang warnanya 'silver' atau 'merah'.

Kesimpulan

Hasil dari perintah tersebut akan mengembalikan semua baris dari tabel "mobil" di mana nilai kolom "warna" adalah 'silver' atau 'merah'. Semua kolom dari baris-baris yang memenuhi kondisi tersebut akan ditampilkan dalam hasilnya

IN +AND

Struktur Query

```
select * from nama_tabel
-> where nama_kolom in ('nilai','nilai2')
-> AND nama_kolom = nilai3
```

Contoh Query

```
select * from mobil
-> where warna IN ('hitam','silver')
-> and harga_rental = 50000;
```

Hasil


```
MariaDB [rental_adel]> select * from mobil
-> where warna IN ('hitam','silver')
-> and harga_rental = 50000;
```

```
+-----+-----+-----+-----+-----+-----+-----+
| id_mobil | no_plat | no_mesin | warna | pemilik | peminjam | harga_rental |
+-----+-----+-----+-----+-----+-----+-----+
|          3 | B 1617 QC | LSQ1112 | SILVER | RANI    | STEA     |          50000 |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.011 sec)
```

Analisis

- `SELECT *`: Bagian kueri ini menentukan bahwa Anda ingin mengambil semua kolom dari `mobil` tabel.
- `FROM mobil`: Bagian kueri ini menentukan tabel tempat Anda ingin mengambil data. Dalam hal ini, Anda ingin mengambil data dari `mobil` tabel.
- `WHERE warna IN ('hitam', 'silver') AND harga_rental = 50000`: Bagian kueri ini menentukan kondisi yang harus dipenuhi data agar dapat diambil. Dalam hal ini, Anda ingin mengambil data yang `warna` (warnanya) adalah 'hitam' atau 'perak' dan `harga_rental` (harga sewa) sama dengan 50.000.

Kesimpulan

perintah tersebut digunakan untuk mencari mobil dengan warna 'hitam' atau 'silver' dan harga sewa sebesar 50.000 dari tabel "mobil".

IN + OR

Struktur Query

```
select * from nama_tabel
-> where nama_kolom in ('nilai','nilai2')
-> OR nama_kolom = nilai3
```

Contoh Query

```
select * from mobil
-> where warna IN ('hitam','silver')
-> and harga_rental = 50000;
```

Hasil

```

MariaDB [rental_adel]> select * from mobil
-> where warna IN ('hitam','silver')
-> or harga_rental = 50000;
+-----+-----+-----+-----+-----+-----+-----+
| id_mobil | no_plat | no_mesin | warna | pemilik | peminjam | harga_rental |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | DD 2650 XY | ACX3560 | HITAM | ADEL | ILA | 50000 |
| 3 | B 1617 QC | LSQ1112 | SILVER | RANI | STEA | 50000 |
| 4 | DD 2210 | UQL1029 | HITAM | RANI | NULL | 150000 |
| 5 | DD 1111 AD | CJH1011 | HITAM | RANI | NULL | 100000 |
+-----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.002 sec)

```

Analisis

1. `warna IN ('hitam', 'silver')`: Kondisi ini akan memfilter baris berdasarkan kolom "warna", memeriksa apakah nilainya adalah "hitam" atau "silver". Perintah ini akan memilih hanya baris-baris di mana warnanya adalah hitam atau silver.
2. `harga_rental = 50000`: Kondisi ini akan memfilter baris berdasarkan kolom "harga_rental", memilih hanya baris-baris di mana harga sewanya adalah 50.000

Kesimpulan

semua baris dari tabel "mobil" yang memiliki warna "hitam" atau "silver" dan harga_rental sebesar 50000.

IN + AND + OPERATOR

Struktur Query

```

select * from nama_tabel
-> where nama_kolom in ('nilai','nilai2')
-> AND nama_kolom > nilai3

```

```

select * from nama_   tabel
-> where nama_kolom in ('nilai','nilai2')
-> AND nama_kolom < nilai3

```

Contoh Query

```

select * from mobil
-> where warna IN ('hitam','silver')
-> and harga_rental > 50000;

```

```
select * from mobil
-> where warna IN ('hitam','silver')
-> and harga_rental < 50000;
```

Hasil

```
MariaDB [rental_adel]> select * from mobil
-> where warna IN ('hitam','silver')
-> AND harga_rental > 50000;
```

id_mobil	no_plat	no_mesin	warna	pemilik	peminjam	harga_rental
4	DD 2210	UQL1029	HITAM	RANI	NULL	150000
5	DD 1111 AD	CJH1011	HITAM	RANI	NULL	100000

2 rows in set (0.001 sec)

```
MariaDB [rental_adel]>
MariaDB [rental_adel]> select * from mobil
-> where warna IN ('hitam','silver')
-> AND harga_rental < 50000;
Empty set (0.001 sec)
```

Analisis

1. `warna IN ('hitam', 'silver')`: Kondisi ini akan memfilter baris berdasarkan kolom "warna", memeriksa apakah nilainya adalah "hitam" atau "silver". Perintah ini akan memilih hanya baris-baris di mana warnanya adalah hitam atau silver.
2. `harga_rental > 50000`: Kondisi ini akan memfilter baris berdasarkan kolom "harga_rental", memilih hanya baris-baris di mana harga sewanya lebih besar dari 50.000.

Kesimpulan

emua baris dari tabel "mobil" yang memiliki warna "hitam" atau "silver" dan harga_rental kurang dari 50000.

LIKE

Mencari Awalan

Struktur Query

```
select * from mobil
```

```
-> where pemilik like 'ad%';
```

Contoh Query

```
select * from [nama_tabel]
-> where [nama_kolom] like 'nama_awal';
```

Hasil

```
MariaDB [rental_adel]> select * from mobil
-> where pemilik like 'ad%';
+-----+-----+-----+-----+-----+-----+-----+
| id_mobil | no_plat   | no_mesin | warna | pemilik | peminjam | harga_rental |
+-----+-----+-----+-----+-----+-----+-----+
| 1        | DD 2650 XY | ACX3560  | HITAN | ADEL    | ILA      | 50000        |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.004 sec)
```

Analisis

`pemilik LIKE 'ad%'` : Kondisi ini akan memfilter baris berdasarkan kolom "pemilik", mencocokkan nilai dengan pola yang diberikan. Dalam hal ini, pola yang diberikan adalah 'ad%', yang berarti mencari nilai pemilik yang dimulai dengan huruf 'ad'. Tanda '%' dalam pola tersebut menunjukkan bahwa ada nol atau lebih karakter setelah 'ad'.

Kesimpulan

semua baris dari tabel "`mobil`" di mana nilai kolom "`pemilik`" dimulai dengan huruf '`ad`'.

Mencari Akhiran

Struktur Query

```
SELECT * FROM mobil
-> WHERE pemilik LIKE '%i';
```

Contoh Query

```
select * from [nama_tabel]
-> where [nama_kolom] like 'nama_akhir';
```

Hasil

```
MariaDB [rental_adel]> select * from mobil  
-> where pemilik like '%i';
```

id_mobil	no_plat	no_mesin	warna	pemilik	peminjam	harga_rental
3	B 1617 QC	LSQ1112	SILVER	RANI	STEAL	50000
4	DD 2210	UQL1029	HITAM	RANI	NULL	150000
5	DD 1111 AD	CJH1011	HITAM	RANI	NULL	100000

3 rows in set (0.289 sec)

Analisis

`pemilik LIKE '%i'` : Kondisi ini akan memfilter baris berdasarkan kolom "pemilik", mencocokkan nilai dengan pola yang diberikan. Dalam hal ini, pola yang diberikan adalah '%i', yang berarti mencari nilai pemilik yang diakhiri dengan huruf 'i'. Tanda '%' di depan pola menunjukkan bahwa ada nol atau lebih karakter sebelum 'i'.

Kesimpulan

Hanya mobil-mobil yang memiliki nilai pada kolom "pemilik" yang diakhiri dengan huruf 'i' yang akan dipilih.

Mencari Awalan & Akhiran

Struktur Query

```
select * from [nama_tabel]  
-> where [nama_kolom] like 'namaawal%namakhir';
```

Contoh Query

```
select * from mobil  
-> where pemilik like 'r%i';
```

Hasil

```
MariaDB [rental_adel]> select * from mobil  
-> where pemilik like 'r%i';
```

id_mobil	no_plat	no_mesin	warna	pemilik	peminjam	harga_rental
3	B 1617 QC	LSQ1112	SILVER	RANI	STEAL	50000
4	DD 2210	UQL1029	HITAM	RANI	NULL	150000
5	DD 1111 AD	CJH1011	HITAM	RANI	NULL	100000

3 rows in set (0.001 sec)

Analisis

- `pemilik LIKE` : Mengindikasikan pencocokan pola pada kolom "pemilik".
- `'r%i'` : Pola yang digunakan untuk pencocokan. Huruf 'r' menunjukkan bahwa nilai kolom "pemilik" harus diawali dengan huruf 'r', sedangkan '%' menunjukkan bahwa setelah huruf 'r', ada nol atau lebih karakter, dan diakhiri dengan huruf 'i'.

Kesimpulan

yang memiliki nilai pada kolom "pemilik" yang diawali dengan huruf 'r', diikuti oleh nol atau lebih karakter, dan diakhiri dengan huruf 'i' yang akan dipilih.

Mencari berdasarkan total karakter

Struktur Query

```
select * from [nama_tabel]
-> where [nama_kolom] like 'awalan___';
```

Contoh Query

```
select * from mobil
-> where pemilik like 'r___';
```

Hasil

```
MariaDB [rental_adel]> select * from mobil
-> where pemilik like 'r___';
```

id_mobil	no_plat	no_mesin	warna	pemilik	peminjam	harga_rental
3	B 1617 QC	LSQ1112	SILVER	RANI	STEA	50000
4	DD 2210	UQL1029	HITAM	RANI	NULL	150000
5	DD 1111 AD	CJH1011	HITAM	RANI	NULL	100000

```
3 rows in set (0.005 sec)
```

Analisis

1. `pemilik LIKE 'r___'` : Kondisi ini akan memfilter baris berdasarkan kolom "pemilik", mencocokkan nilai dengan pola yang diberikan. Dalam hal ini, pola yang diberikan adalah 'r___', yang berarti mencari nilai pemilik yang terdiri dari lima karakter, di mana karakter pertama adalah 'r' dan tiga karakter berikutnya boleh apa saja.

Kesimpulan

digunakan untuk mencari total karakter

Kombinasi

Struktur Query

```
select * from [nama_tabel]
-> where [nama_kolom] like '___akhiran%';
```

Contoh Query

```
select * from mobil
-> where pemilik like '___i%';
```

Hasil

```
MariaDB [sekolah]> select * from pelanggan
-> where nama_mobil like '___l%';
+-----+-----+-----+
| nama_mobil | plat_mobil | warna_mobil |
+-----+-----+-----+
| adel       | hrv        | 081243935227 |
| aril       | haq        | 0823456280   |
+-----+-----+-----+
2 rows in set (0.040 sec)
```

Analisis

pemilik LIKE '___i%': Kondisi ini akan memfilter baris berdasarkan kolom "pemilik", mencocokkan nilai dengan pola yang diberikan. Dalam hal ini, pola yang diberikan adalah 'i%', **yang berarti mencari nilai pemilik yang terdiri dari setidaknya empat karakter, di mana karakter ketiga adalah 'i', dan karakter-karakter berikutnya boleh apa saja**

Kesimpulan

digunakan untuk mencari akhiran nama yang berakhiran sesuai huruf yang ingin di cari

Not like

Struktur Query

```
select * from [nama_tabel] where [nama_kolom] not like 'awalan%';
```

Contoh Query

```
select * from mobil where peminjam not like 'r%';
```

Hasil

```
MariaDB [rental_adel]> select * from mobil where peminjam not like 'r%';
+-----+-----+-----+-----+-----+-----+-----+
| id_mobil | no_plat | no_mesin | warna | pemilik | peminjam | harga_rental |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | DD 2650 XY | ACX3560 | HITAN | ADEL | ILA | 50000 |
| 2 | DD 2440 AX | BCS1120 | MERAH | ALYA | IRA | 100000 |
| 3 | B 1617 QC | LSQ1112 | SILVER | RANI | STEA | 50000 |
+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.001 sec)
```

Analisis

1. **SELECT * FROM mobil:** Ini akan memilih semua kolom dari tabel `mobil`.
2. **WHERE peminjam NOT LIKE 'r%':** Kondisi ini akan menyaring baris di mana nilai kolom `peminjam` tidak dimulai dengan huruf 'r'. Operator `LIKE` digunakan untuk pattern matching, dan `'r%'` berarti string yang dimulai dengan huruf 'r'.

Kesimpulan

NULL & NOT NULL

Mencari data kosong

Struktur Query

```
SELECT * FROM [namatabel] WHERE [namakolom] IS NULL;
```

Contoh Query

```
SELECT * FROM mobil WHERE peminjam IS NULL;
```

Hasil


```
MariaDB [rental_adel]> SELECT * FROM mobil WHERE peminjam IS NULL;
```

id_mobil	no_plat	no_mesin	warna	pemilik	peminjam	harga_rental
4	DD 2210	UQL1029	HITAM	RANI	NULL	150000
5	DD 1111 AD	CJH1011	HITAM	RANI	NULL	100000

```
2 rows in set (0.076 sec)
```

Analisis

- **SELECT** : Digunakan untuk memilih kolom atau data yang ingin ditampilkan dalam hasil query.
- **Tanda asterisk (*)** digunakan sebagai wildcard dalam SQL, yang berarti "semua kolom". Dalam konteks ini, "*" menunjukkan bahwa semua kolom dalam tabel "mobil" akan ditampilkan dalam hasil query.
- **FROM** : Digunakan untuk menentukan tabel sumber data dari mana data akan diambil. Dalam hal ini, tabel sumber data adalah "mobil".
- **mobil** : Nama tabel yang spesifik di mana data akan diambil.
- **WHERE** : Digunakan untuk memberikan kriteria atau kondisi yang harus dipenuhi untuk mengambil data yang relevan. Dalam kasus ini, kondisi adalah "peminjam IS NULL", yang berarti hanya baris dengan kolom "peminjam" yang memiliki nilai **NULL** yang akan dimasukkan dalam hasil query.

Kesimpulan

Perintah **SELECT *** digunakan untuk memilih semua kolom dari tabel 'mobil'. Kemudian, **FROM mobil** menunjukkan tabel yang digunakan untuk mengambil data, yaitu tabel 'mobil'. Selanjutnya, **WHERE peminjam IS NULL** menggunakan klausa **WHERE** untuk memfilter baris-baris yang akan diambil. Kondisi **peminjam IS NULL** digunakan untuk memeriksa apakah nilai kolom 'peminjam' adalah **NULL**. Operator **IS NULL** digunakan untuk memeriksa apakah nilai kolom adalah **NULL**. Dengan demikian, query ini akan mengembalikan baris-baris di mana kolom 'peminjam' memiliki nilai **NULL**.

Mencari data yang tidak kosong

Struktur Query

```
SELECT * FROM [namatabel] WHERE [namakolom] IS NOT NULL;
```

Contoh Query

```
SELECT * FROM mobil WHERE peminjam IS NOT NULL;
```

Hasil

```
MariaDB [rental_adel]> SELECT * FROM mobil WHERE peminjam IS NOT NULL;
```

id_mobil	no_plat	no_mesin	warna	pemilik	peminjam	harga_rental
1	DD 2650 XY	ACX3560	HITAN	ADEL	ILA	50000
2	DD 2440 AX	BCS1120	MERAH	ALYA	IRA	100000
3	B 1617 QC	LSQ1112	SILVER	RANI	STEAL	50000

3 rows in set (0.090 sec)

Analisis

- **SELECT** : Digunakan untuk memilih kolom atau data yang ingin ditampilkan dalam hasil query.
- ***** : Tanda (*) digunakan sebagai wildcard dalam SQL, yang berarti "semua kolom". Dalam konteks ini, "*" menunjukkan bahwa semua kolom dalam tabel "mobil" akan ditampilkan dalam hasil query.
- **FROM** : Digunakan untuk menentukan tabel sumber data dari mana data akan diambil. Dalam hal ini, tabel sumber data adalah "mobil".
- **mobil** : Nama tabel yang spesifik di mana data akan diambil.
- **WHERE** : Digunakan untuk memberikan kriteria atau kondisi yang harus dipenuhi untuk mengambil data yang relevan. Dalam kasus ini, kondisi adalah "peminjam IS NOT NULL", yang berarti hanya baris dengan kolom "peminjam" yang memiliki nilai tidak NULL yang akan dimasukkan dalam hasil query.

Kesimpulan

Perintah SQL `SELECT * FROM mobil WHERE peminjaman IS NOT NULL;` digunakan untuk mengambil semua baris dari tabel "mobil" di mana kolom "peminjaman" memiliki nilai yang tidak NULL. Dengan kata lain, perintah ini mengambil data mobil yang sedang dipinjam atau sudah dipinjam dari tabel tersebut.

ORDEY BY & LIMIT 8

Mengurutkan data dari data terkecil

Struktur Query

```
SELECT * FROM [namatabel] ORDER BY namakolom ASC;
```

Contoh Query

```
SELECT * FROM mobil ORDER BY pemilik ASC;
```

Hasil

```
MariaDB [rental_ade1]> SELECT * FROM mobil ORDER BY pemilik ASC;
```

id_mobil	no_plat	no_mesin	warna	pemilik	peminjam	harga_rental
1	DD 2650 XY	ACX3560	HITAM	ADEL	ILA	50000
2	DD 2440 AX	BCS1120	MERAH	ALYA	IRA	100000
3	B 1617 QC	LSQ1112	SILVER	RANI	STEPA	50000
4	DD 2210	UQL1029	HITAM	RANI	NULL	150000
5	DD 1111 AD	CJH1011	HITAM	RANI	NULL	100000

5 rows in set (0.124 sec)

Analisis

- **SELECT:** Digunakan untuk memilih kolom atau data yang ingin ditampilkan dalam hasil query.
- **:** *Tanda asterisk (*)* digunakan sebagai wildcard dalam SQL, yang berarti "semua kolom". Dalam konteks ini, "*" menunjukkan bahwa semua kolom dalam tabel "mobil" akan ditampilkan dalam hasil query.
- **FROM:** Digunakan untuk menentukan tabel sumber data dari mana data akan diambil. Dalam hal ini, tabel sumber data adalah "mobil".
- **mobil:** Nama tabel yang spesifik di mana data akan diambil.
- **ORDER BY:** Digunakan untuk mengurutkan hasil query berdasarkan kolom tertentu.
- **pemilik:** Nama kolom yang akan digunakan sebagai kriteria pengurutan. Dalam kasus ini, kolom "pemilik" akan digunakan.
- **ASC:** Singkatan dari "ascending" yang menandakan pengurutan secara menaik (dari nilai terkecil ke nilai terbesar).

Kesimpulan

Perintah SQL ini memberikan hasil berupa semua data yang ada dalam tabel "mobil", diurutkan berdasarkan kolom "pemilik" secara menaik. Hasilnya akan mengembalikan semua baris dalam tabel "mobil" dalam urutan yang diinginkan.

Mengurutkan data dari data terbesar

Struktur Query

```
SELECT * FROM [namatabel] ORDER BY namakolom ASC;
```

Contoh Query

```
SELECT * FROM mobil ORDER BY peminjam DESC;
```

Hasil

```
MariaDB [rental_adel]> SELECT * FROM mobil ORDER BY peminjam DESC;
```

id_mobil	no_plat	no_mesin	warna	pemilik	peminjam	harga_rental
3	B 1617 QC	LSQ1112	SILVER	RANI	STEAL	50000
2	DD 2440 AX	BCS1120	MERAH	ALYA	IRA	100000
1	DD 2650 XY	ACX3560	HITAM	ADEL	ILA	50000
4	DD 2210	UQL1029	HITAM	RANI	NULL	150000
5	DD 1111 AD	CJH1011	HITAM	RANI	NULL	100000

```
5 rows in set (0.002 sec)
```

Analisis

- **SELECT:** Digunakan untuk memilih kolom atau data yang ingin ditampilkan dalam hasil query.
- **:** *Tanda asterisk (*)* digunakan sebagai wildcard dalam SQL, yang berarti "semua kolom". Dalam konteks ini, "*" menunjukkan bahwa semua kolom dalam tabel "mobil" akan ditampilkan dalam hasil query.
- **FROM:** Digunakan untuk menentukan tabel sumber data dari mana data akan diambil. Dalam hal ini, tabel sumber data adalah "mobil".
- **mobil:** Nama tabel yang spesifik di mana data akan diambil.
- **ORDER BY:** Digunakan untuk mengurutkan hasil query berdasarkan kolom tertentu.
- **peminjam:** Nama kolom yang akan digunakan sebagai kriteria pengurutan. Dalam kasus ini, kolom "peminjam" akan digunakan.
- **DESC:** Singkatan dari "descending" yang menandakan pengurutan secara menurun (dari nilai terbesar ke nilai terkecil).

Kesimpulan

Perintah SQL ini menghasilkan semua data yang ada dalam tabel "mobil" dan mengurutkannya secara menurun berdasarkan kolom "peminjam". Urutan menurun (descending) ditentukan oleh kata kunci "DESC" yang digunakan dalam perintah "ORDER BY".

Membatasi data yang tampil

```
select * from mobil where warna = "hitam" order by harga asc limit 2;
```

DISTINCT

Struktur Query

```
SELECT DISTINCT(namakolom) FROM namatabel;
```

```
SELECT DISTINCT(namakolom) FROM namatabel ORDER BY namakolom DESC;
```

Contoh Query

```
SELECT DISTINCT(pemilik) FROM mobil;
```

```
SELECT DISTINCT(harga_rental) FROM mobil ORDER BY harga_rental DESC;
```

Hasil

```
MariaDB [rental_adel]> SELECT DISTINCT(pemilik) FROM mobil;
+-----+
| pemilik |
+-----+
| ADEL    |
| ALYA    |
| RANI    |
+-----+
3 rows in set (0.185 sec)
```

```
MariaDB [rental_adel]> SELECT DISTINCT(harga_rental) FROM mobil ORDER BY harga_rental DESC;
+-----+
| harga_rental |
+-----+
| 150000       |
| 100000       |
| 50000        |
+-----+
3 rows in set (0.001 sec)
```

Analisis

1. `SELECT DISTINCT(harga_rental)`: Perintah ini mengambil nilai unik dari kolom "harga_rental" dalam tabel "mobil". Klausula `DISTINCT` digunakan untuk memastikan bahwa hanya nilai unik yang diambil, sehingga tidak ada duplikasi dalam hasil.
2. `FROM mobil`: Ini menunjukkan bahwa tabel yang digunakan dalam perintah ini adalah "mobil". Pastikan tabel "mobil" ada dalam database yang aktif.
3. `ORDER BY harga_rental DESC`: Ini adalah klausa yang digunakan untuk mengurutkan nilai unik "harga_rental" secara menurun (descending). `DESC` adalah kata kunci yang menunjukkan urutan menurun.

Kesimpulan

Perintah SQL ini memberikan hasil berupa nilai unik dari kolom "harga_rental" dalam tabel "mobil", diurutkan secara menurun. Hasilnya akan mengembalikan daftar harga rental yang unik dalam urutan menurun.

CONCAT, CONCAT_WS, AS

Menggabungkan kolom tanpa pemisah

Struktur Query

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

Contoh Query

```
SELECT CONCAT(pemilik, warna) FROM daftar_mobil;
```

Hasil

```
MariaDB [rental_adel]> SELECT CONCAT(pemilik, warna) FROM daftar_mobil;
+-----+
| CONCAT(pemilik, warna) |
+-----+
| ADELHITAN              |
| ALYAMERAH             |
| RANISILVER             |
| RANIHITAM              |
| RANIHITAM              |
| QIQIMERAH              |
+-----+
6 rows in set (0.402 sec)
```

Analisis

1. `SELECT CONCAT(pemilik, warna)`: Perintah ini menggunakan fungsi `CONCAT` untuk menggabungkan nilai dari kolom "pemilik" dan "warna" dalam tabel "daftar_mobil". Fungsi `CONCAT` digunakan untuk menggabungkan dua atau lebih string menjadi satu string.
2. `FROM daftar_mobil`: Ini menunjukkan bahwa tabel yang digunakan dalam perintah ini adalah "daftar_mobil". Pastikan tabel "daftar_mobil" ada dalam database yang aktif.

Kesimpulan

Perintah SQL ini menghasilkan hasil berupa string yang merupakan hasil penggabungan nilai dari kolom "pemilik" dan "warna" dalam tabel "daftar_mobil".

Menggabungkan kolom dengan pemisah

Struktur Query

```
SELECT CONCAT_WS("-", nama2_kolom) FROM nama_tabel;
```

Contoh Query

```
SELECT CONCAT_WS("-", no_plat, no_mesin, id_mobil) FROM daftar_mobil;
```

Hasil

```
MariaDB [rental_adel]> SELECT CONCAT_WS("-", no_plat, no_mesin, id_mobil) FROM daftar_mobil;
+-----+
| CONCAT_WS("-", no_plat, no_mesin, id_mobil) |
+-----+
| DD 2650 XY-ACX3560-1                        |
| DD 2440 AX-BCS1120-2                        |
| B 1617 QC-LSQ1112-3                         |
| DD 2210-UQL1029-4                          |
| DD 1111 AD-CJH1011-5                       |
| B 127 N-AESC78-7                           |
+-----+
6 rows in set (0.179 sec)
```

Analisis

1. `SELECT CONCAT_WS("-", no_plat, no_mesin, id_mobil)`: Perintah ini menggunakan fungsi `CONCAT_WS` untuk menggabungkan nilai dari kolom "no_plat", "no_mesin", dan "id_mobil" dalam tabel "daftar_mobil". Fungsi `CONCAT_WS` (Concatenate With Separator) menggabungkan nilai-nilai string dengan memasukkan pemisah yang ditentukan di antara mereka.
2. `FROM daftar_mobil`: Ini menunjukkan bahwa tabel yang digunakan dalam perintah ini adalah "daftar_mobil". Pastikan tabel "daftar_mobil" ada dalam database yang aktif.

Kesimpulan

Perintah SQL ini menghasilkan hasil berupa string yang merupakan hasil penggabungan nilai dari kolom "no_plat", "no_mesin", dan "id_mobil" dalam tabel "daftar_mobil", dengan pemisah tanda hubung ("-") di antara mereka.

Memberikan nama kolom alias

Struktur Query

```
SELECT CONCAT_WS("+",nama2_kolom) AS nama_kolom_baru FROM nama_tabel;
```

Contoh Query

```
SELECT CONCAT_WS("+",pemilik,peminjaman) AS COLLAB FROM mobil;
```

Hasil

```
MariaDB [rental_adel]> SELECT CONCAT_WS("+",pemilik,peminjam) AS COLLAB FROM daftar_mobil;
+-----+
| COLLAB |
+-----+
| ADEL+anil |
| ALYA+ira  |
| RANI+sisil |
| RANI      |
| RANI      |
| QIQI      |
+-----+
6 rows in set (0.001 sec)
```

Analisis

1. `SELECT CONCAT_WS("+", pemilik, peminjaman) AS COLLAB`: Perintah ini menggunakan fungsi `CONCAT_WS` untuk menggabungkan nilai dari kolom "pemilik" dan "peminjaman" dalam tabel "mobil". Fungsi `CONCAT_WS` (Concatenate With Separator) menggabungkan nilai-nilai string dengan memasukkan pemisah yang ditentukan di antara mereka. Dalam hal ini, pemisah yang digunakan adalah tanda tambah ("+"). Alias "COLLAB" digunakan untuk memberi nama kolom hasil penggabungan.
2. `FROM mobil`: Ini menunjukkan bahwa tabel yang digunakan dalam perintah ini adalah "mobil". Pastikan tabel "mobil" ada dalam database yang aktif.

Kesimpulan

Perintah SQL ini menghasilkan hasil berupa kolom baru yang bernama "COLLAB", yang berisi string hasil penggabungan nilai dari kolom "pemilik" dan "peminjaman" dalam tabel "mobil", dengan pemisah tanda tambah ("+") di antara mereka.

VIEW

Membuat tabel virtual

Struktur Query


```
CREATE VIEW nama_kolom_baru AS  
-> SELECT nama2_kolom  
-> FROM nama_tabel;  
-> WHERE nama_kolom = isi_kolom;
```

Contoh Query

```
CREATE VIEW info_no_platt AS  
-> SELECT id_mobil, no_plat, pemilik, peminjaman  
-> FROM daftar_mobil  
-> WHERE pemilik = "ADEL";
```

Hasil

```
MariaDB [rental_adel]> CREATE VIEW info_no_platt AS  
-> SELECT id_mobil, no_plat, pemilik, peminjam  
-> from daftar_mobil  
-> where pemilik = "ADEL";  
Query OK, 0 rows affected (0.077 sec)
```

Analisis

1. CREATE VIEW info_no_platt: Ini adalah perintah untuk membuat view baru dengan nama "info_no_platt". View adalah objek database yang berfungsi sebagai tabel virtual yang terdiri dari hasil query yang ditentukan.
2. SELECT id_mobil, no_plat, pemilik, peminjaman: Ini adalah perintah SELECT yang digunakan dalam pembuatan view. Perintah ini menentukan kolom mana yang akan dipilih dari tabel "daftar_mobil" untuk dimasukkan ke dalam view "info_no_platt".
3. FROM daftar_mobil: Ini menunjukkan bahwa view "info_no_platt" akan dibuat berdasarkan data yang ada di tabel "daftar_mobil". Pastikan tabel "daftar_mobil" ada dalam database yang aktif.
4. WHERE pemilik = 'ADEL': Ini adalah klausa WHERE yang digunakan untuk memberikan kondisi untuk memilih hanya baris yang memiliki nilai pemilik sama dengan 'ADEL'. Hal ini akan memfilter data yang dimasukkan ke dalam view sehingga hanya data dengan pemilik "ADEL" yang akan ditampilkan.

Kesimpulan

Perintah SQL ini membuat sebuah view baru dengan nama "info_no_platt" yang berisi data dari tabel "daftar_mobil" dengan kondisi bahwa pemilik mobil adalah "ADEL". View ini akan berisi kolom "id_mobil", "no_plat", "pemilik", dan "peminjaman" untuk baris-baris yang memenuhi kondisi tersebut.

Menampilkan tabel virtual

Struktur Query

```
SELECT * FROM nama_tabel_baru b ;
```

Contoh Query

```
SELECT * FROM info_no_plat;
```

Hasil

```
MariaDB [rental_adel]> SELECT * FROM info_no_platt;
+-----+-----+-----+-----+
| id_mobil | no_plat | pemilik | peminjam |
+-----+-----+-----+-----+
|          1 | DD 2650 XY | ADEL    | anil     |
+-----+-----+-----+-----+
1 row in set (0.166 sec)
```

Analisis

1. SELECT : *Perintah ini digunakan untuk memilih semua kolom yang ada dalam view "info_no_plat". Tanda "" menunjukkan bahwa semua kolom akan dipilih.*
2. FROM info_no_plat: *Ini menunjukkan bahwa data akan diambil dari view "info_no_plat". Pastikan view "info_no_plat" sudah dibuat sebelumnya.*

Kesimpulan

Perintah SQL ini mengambil semua data yang ada dalam view "info_no_plat" dan mengembalikannya sebagai hasil query.

Menghapus tabel virtual

Struktur Query

```
DROP VIEW nama_tabel_baru;
```

Contoh Query

```
DROP VIEW info_no_plat;
```

Hasil

```
MariaDB [rental_adel]> DROP VIEW info_no_platt;  
Query OK, 0 rows affected (0.001 sec)
```

Analisis

DROP VIEW info_no_plat: Ini adalah perintah untuk menghapus view dengan nama "info_no_plat" dari database. Perintah DROP VIEW digunakan untuk menghapus view yang telah dibuat sebelumnya.

Kesimpulan

Perintah SQL ini menghapus view "info_no_plat" dari database.

Tantangan View

Nomor 1

Penjelasan

CREATE VIEW mobil_tanpa_peminjam AS : adalah perintah untuk membuat sebuah view baru atau seperti tabel baru dalam basis data dengan nama mobil_tanpa_peminjam.

- `SELECT no_plat, peminjaman` : adalah perintah untuk memilih dua kolom, yaitu `no_plat` dan `peminjam`, dari tabel `mobil`.
- `FROM mobil` : Menunjukkan bahwa data diambil dari tabel bernama mobil.
- `WHERE peminjam IS NULL` : adalah klausa `WHERE` yang mencari baris-baris dari tabel mobil dimana nilai kolom peminjam adalah `NULL`.
- `SELECT *` : adalah perintah untuk memilih semua kolom dari view atau tabel.
- `FROM mobil_Tanpa_peminjam` : Menunjukkan bahwa data diambil dari view yang disebut mobil_Tanpa_peminjam, yang telah dibuat sebelumnya.

Query

```
CREATE VIEW  
-> mobil_tanpa_peminjam AS  
-> SELECT no_plat,peminjaman  
-> FROM mobil  
-> WHERE peminjaman IS NULL;
```

Hasil

```
MariaDB [rental_adel]> create view
-> mobil_tanpa_peminjam AS
-> SELECT no_plat,peminjam
-> FROM daftar_mobil
-> WHERE peminjam IS NULL;
Query OK, 0 rows affected (0.048 sec)
```

Kesimpulan

`CREATE VIEW mobil_tanpa_peminjam AS Select no_plat, peminjaman FROM mobil WHERE peminjaman IS NULL;` digunakan untuk membuat sebuah view baru bernama `mobil_Tanpa_peminjam`. Viewnya berisi dua kolom, yaitu `no_plat` dan `peminjaman`, yang diambil dari tabel `mobil` hanya baris-baris yang memiliki nilai `NULL` pada kolom `peminjam` yang dimasukkan ke dalam view.

`SELECT * FROM mobil_tanpa_peminjam;` digunakan untuk menampilkan semua data dari view `mobil_Tanpa_peminjam`, yang telah dibuat sebelumnya dengan kriteria yang bernilai `NULL`.

Nomor 2

Penjelasan

- `UPDATE mobil` : adalah perintah untuk memperbarui data dalam tabel yang disebut mobil.
- `SET peminjaman = NULL` : menetapkan nilai kolom `peminjam` menjadi `NULL`.
- `WHERE peminjam= 'ADEL'` : adalah klausa `WHERE` yang membatasi update hanya pada baris-baris dimana nilai kolom `peminjam` adalah 'ADEL'. Maksudnya perubahan hanya akan berlaku untuk baris-baris yang memiliki `peminjam` dengan nama 'ADEL'.
- `SELECT *` : adalah perintah untuk memilih semua kolom dari view atau tabel.
- `FROM mobil_tanpa_peminjam` : Menunjukkan bahwa data diambil dari view yang disebut "mobil_tanpa_peminjam", yang telah dibuat sebelumnya.

Query

```
UPDATE mobil
-> SET peminjaman = NULL
-> WHERE peminjaman = 'ADEL';
```

Hasil

```
MariaDB [mobil]> SELECT * FROM MOBIL;
```

id_mobil	no_plat	no_mesin	warna	pemilik	peminjam	harga_rental
1	DD 1105 AD	VXCFB	HITAM	ADEL		120000

1 row in set (0.001 sec)

Kesimpulan

`UPDATE mobil SET peminjaman = NULL WHERE peminjaman = 'ADEL';` nilai pada kolom peminjaman pada tabel `mobil` yang memiliki nilai 'ADEL' akan diubah menjadi NULL.

Kesimpulannya, perintah digunakan untuk menghapus atau mengubah nilai peminjaman menjadi NULL untuk semua data di tabel mobil yang berada di kolom `peminjaman` memiliki nilai 'ADEL'.

`SELECT * FROM mobil_tanpa_peminjam;` digunakan untuk menampilkan semua data dari view `mobil_tanpa_peminjam`, yang telah dibuat sebelumnya dengan mengubah atau menghapus nilai peminjam menjadi NULL untuk tabel mobil dimana peminjam memiliki nilai ADEL.

Nomor 3

View digunakan untuk menyaring data sesuai dengan kriteria tertentu, seperti menampilkan data yang memiliki nilai NULL pada kolom tertentu atau mengubah salah satu data peminjaman menjadi NULL.

Memberikan pandangan yang jelas tentang mobil yang tersedia untuk disewakan atau yang belum memiliki peminjam.

Dengan membuat view, kita dapat membatasi akses ke data sensitif atau kolom tertentu dari tabel yang mungkin tidak perlu diakses oleh semua pengguna.

Dengan membuat view untuk kueri yang sering digunakan, Anda dapat menghindari pengulangan kode SQL yang sama di beberapa tempat dalam aplikasi atau prosedur penyimpanan.

AGREGASI

SUM

Penjelasan

- `SELECT SUM(harga_rental)` : Fungsi `SUM()` digunakan untuk menghitung total atau jumlah dari semua nilai dalam kolom `harga_rental`.
- `FROM mobil` : Menentukan tabel mobil dari mana data akan diambil.

Struktur Query

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

Contoh Query

```
SELECT SUM(harga_rental) FROM mobil;
```

Hasil

```
MariaDB [rental_alya]> SELECT SUM(harga_rental) FROM daftar_mobil;
+-----+
| SUM(harga_rental) |
+-----+
|           600000 |
+-----+
1 row in set (0.110 sec)
```

Kesimpulan

Query `SELECT SUM(harga_rental) FROM mobil;` akan menghitung dan menampilkan total atau jumlah keseluruhan dari nilai-nilai yang terdapat pada kolom `harga_rental` di dalam tabel `mobil`. Hasil query ini akan memberikan informasi total atau keseluruhan harga rental untuk semua mobil yang ada di dalam tabel tersebut.

Count

Penjelasan

PENJELASAN 1

- `SELECT COUNT(pemilik)` - Fungsi `COUNT()` digunakan untuk menghitung jumlah baris atau record yang ada di dalam kolom `pemilik`.
- `FROM mobil` - Menentukan tabel `mobil` dari mana data akan diambil.

PENJELASAN 2

- `SELECT COUNT(peminjaman)` - Fungsi `COUNT()` digunakan untuk menghitung jumlah baris atau record yang ada di dalam kolom `peminjaman`.
- `FROM mobil` - Menentukan tabel `mobil` dari mana data akan diambil.

Struktur Query

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

Contoh Query

```
SELECT COUNT(pemilik) FROM mobil;  
SELECT COUNT(peminjaman) FROM mobil;
```

Hasil

```
MariaDB [rental_alya]> SELECT COUNT(pemilik) FROM daftar_mobil;  
+-----+  
| COUNT(pemilik) |  
+-----+  
|                7 |  
+-----+  
1 row in set (0.001 sec)
```

Kesimpulan

KESIMPULAN 1

Query `SELECT COUNT(pemilik) FROM mobil;` akan menghitung dan menampilkan jumlah total pemilik mobil yang ada di dalam tabel mobil. Hasil query ini akan memberikan informasi tentang berapa banyak pemilik mobil yang tercatat di dalam tabel tersebut.

KESIMPULAN 2

Query `SELECT COUNT(peminjaman) FROM mobil;` akan menghitung dan menampilkan jumlah total peminjaman mobil yang ada di dalam tabel mobil. Hasil query ini akan memberikan informasi tentang berapa banyak peminjaman mobil yang tercatat di dalam tabel tersebut.

MIN

Penjelasan

- `SELECT MIN(harga_rental)` : Fungsi `MIN()` digunakan untuk mencari nilai minimum (terkecil) dari kolom `harga_rental`.
- `AS MINIMAL` : Hasil dari fungsi `MIN()` akan ditampilkan dengan nama alias "MINIMAL".
- `FROM mobil` : Menentukan tabel mobil dari mana data akan diambil.

Struktur Query

```
SELECT MIN(nama_kolom) AS nilai_minimum FROM nama_tabel;
```

Contoh Query

```
SELECT MIN(harga_rental) AS MINIMAL FROM mobil;
```

Hasil

```
MariaDB [rental_alya]> SELECT MIN(harga_rental) AS MINIMAL FROM daftar_mobil;
+-----+
| MINIMAL |
+-----+
| 50000   |
+-----+
1 row in set (0.000 sec)
```

Kesimpulan

Query `SELECT MIN(harga_rental) AS MINIMAL FROM mobil;` akan mencari dan menampilkan harga rental mobil yang paling murah atau terkecil dari semua data yang ada di dalam tabel mobil. Hasil query ini akan memberikan informasi tentang harga rental mobil terendah yang tercatat di dalam tabel tersebut.

MAX

Penjelasan

- `SELECT MAX(harga_rental)` : Fungsi `MAX()` digunakan untuk mencari nilai maksimum (terbesar) dari kolom `harga_rental`.
- `AS MAXIMAL` : Hasil dari fungsi `MAX()` akan ditampilkan dengan nama alias "MAXIMAL".
- `FROM mobil` : Menentukan tabel mobil dari mana data akan diambil.

Struktur Query

```
SELECT MAX(nama_kolom) AS nilai_minimum FROM nama_tabel;
```

Contoh Query

```
SELECT MAX(harga_rental) AS MAXIMAL FROM mobil;
```

Hasil

```
MariaDB [rental_alya]> SELECT MAX(harga_rental) AS MAXIMAL FROM daftar_mobil;
+-----+
| MAXIMAL |
+-----+
| 150000   |
+-----+
1 row in set (0.001 sec)
```


Kesimpulan

Query `SELECT MAX(harga_rental) AS MAXIMAL FROM mobil;` akan mencari dan menampilkan harga rental mobil yang paling mahal atau terbesar dari semua data yang ada di dalam tabel mobil. Hasil query ini akan memberikan informasi tentang harga rental mobil tertinggi yang tercatat di dalam tabel tersebut.

AVG

Penjelasan

- `SELECT AVG(harga_rental)` : Fungsi `AVG()` digunakan untuk menghitung rata-rata (average) dari nilai-nilai pada kolom `harga_rental`.
- `AS RATA_RATA` : Hasil dari fungsi `AVG()` akan ditampilkan dengan nama alias "`RATA_RATA`".
- `FROM mobil` : Menentukan tabel mobil dari mana data akan diambil.

Struktur Query

```
SELECT AVG(nama_kolom) AS rata_rata FROM nama_tabel;
```

Contoh Query

```
SELECT AVG(harga_rental) AS RATA_RATA FROM mobil;
```

Hasil

```
MariaDB [rental_alya]> SELECT AVG(harga_rental) AS RATA_RATA FROM daftar_mobil;
+-----+
| RATA_RATA |
+-----+
| 100000.0000 |
+-----+
1 row in set (0.001 sec)
```

Kesimpulan

Query `SELECT AVG(harga_rental) AS RATA_RATA FROM mobil;` akan menghitung dan menampilkan rata-rata (average) harga rental mobil dari semua data yang ada di dalam tabel mobil. Hasil query ini akan memberikan informasi tentang harga rental mobil rata-rata yang tercatat di dalam tabel tersebut.

TANTANGAN GROUP BY HAVING

1.tampilkan jumlah data mobil dan kelompok kan berdasarkan warna nya sesuai dengan tabel mobil kalian.

Struktur Query

```
select nama_data, COUNT(nama_data) AS nama_sementara FROM nama_tabel GROUP BY nama_data;
```

Query

```
select warna, COUNT(id_mobil) AS Jumlah_Data_Mobil FROM data_mobil GROUP BY warna;
```

Hasil

```
MariaDB [rental_adels]> select warna, COUNT(id_mobil) AS Jumlah_Data_Mobil FROM mobil GROUP BY warna;
```

warna	Jumlah_Data_Mobil
Hitam	5
Merah	1
Silver	1

```
3 rows in set (0.018 sec)
```

Analisis

- **SELECT kLausa :** warna: Memilih kolom warna dari tabel data_mobil.
- **``COUNT(id_mobil) AS Jumlah_Data_Mobil:** Menghitung jumlah baris (mobil) untuk setiap warna unik dan memberi alias Jumlah_Data_Mobil pada hasil hitungan tersebut.
- **FROM kLausa:** data_mobil: Menentukan tabel data_mobil sebagai sumber data.
- **GROUP BY kLausa:** warna: Mengelompokkan hasil query berdasarkan nilai di kolom warna. Setiap nilai unik dalam kolom warna akan menjadi satu grup.

Kesimpulan

1. Mengelompokkan Data Berdasarkan Warna: Data dalam tabel data_mobil dikelompokkan berdasarkan kolom warna.
2. Menghitung Jumlah Mobil untuk Setiap Warna: Menggunakan fungsi COUNT(id_mobil) untuk menghitung jumlah mobil dalam setiap grup warna.
3. Memberikan Hasil yang Jelas: Hasil dari query ini menunjukkan jumlah mobil untuk setiap warna dalam tabel data_mobil, dengan kolom Jumlah_Data_Mobil menunjukkan hitungan tersebut.

2.berdasarkan query ini tampilkan yang lebih BESAR dari 3 atau sama dengan 3 pemilik mobil nya

Struktur Query

```
select nama_data, COUNT(nama_data) AS nama_sementara from nama_tabel GROUP BY  
nama_data HAVING COUNT(nama_data) >= 3;
```

Query

```
select pemilik, COUNT(id_mobil) AS jumlah_mobil from data_mobil GROUP BY pemilik  
HAVING COUNT(id_mobil) >= 3;
```

Hasil

```
MariaDB [rental_adels]> select pemilik, COUNT(id_mobil) AS jumlah_mobil from mobil GROUP BY pemilik HAVING COUNT(id_mobil) >= 3;  
+-----+-----+  
| pemilik | jumlah_mobil |  
+-----+-----+  
| Ibe     | 3            |  
+-----+-----+  
1 row in set (0.001 sec)
```

Analisis

1. SELECT Klausa pemilik: Kolom ini dipilih dari tabel data_mobil. Kolom pemilik berisi data tentang pemilik mobil.
2. COUNT(id_mobil) AS jumlah_mobil: Fungsi agregat COUNT digunakan untuk menghitung jumlah baris dalam setiap grup yang memiliki pemilik yang sama. Hasil hitungan ini diberi alias jumlah_mobil, sehingga dalam hasil akhir, kolom ini akan diberi nama jumlah_mobil.
3. FROM Klausa data_mobil: Tabel ini merupakan sumber data dari query. Tabel ini diasumsikan berisi data mobil, termasuk kolom pemilik dan id_mobil.
4. GROUP BY pemilik: Pernyataan ini mengelompokkan baris-baris data berdasarkan nilai dalam kolom pemilik. Semua baris yang memiliki nilai pemilik yang sama akan dimasukkan ke dalam grup yang sama.
5. HAVING COUNT(id_mobil) >= 3: Pernyataan ini menyaring grup-grup yang terbentuk berdasarkan hasil agregat. Hanya grup yang memiliki jumlah baris (mobil) setidaknya 3 yang akan dimasukkan dalam hasil akhir. HAVING digunakan setelah pengelompokan data, berbeda dengan WHERE yang digunakan sebelum pengelompokan.

Kesimpulan

1. Mengelompokkan Data Berdasarkan Pemilik: Data dalam tabel data_mobil dikelompokkan berdasarkan kolom pemilik.
2. Menghitung Jumlah Mobil untuk Setiap Pemilik: Menggunakan fungsi COUNT(id_mobil) untuk menghitung jumlah mobil dalam setiap grup pemilik. Hasil hitungan ini diberi alias jumlah_mobil.
3. Menyaring Grup dengan Klausa HAVING: Menggunakan klausa HAVING untuk menyaring dan hanya menampilkan grup yang memiliki jumlah mobil (baris) setidaknya 3.

3.tampilkan semua pemilik dengan jumlah mobilnya yang memiliki atau sama dengan 3 mobil

Struktur Query

```
SELECT nama_data, COUNT(nama_data) AS nama_sementara FROM nama_tabel GROUP BY nama_data;
```

Query

```
SELECT pemilik,  
COUNT(id_mobil) AS jumlah_mobil  
FROM data_mobil GROUP BY pemilik;
```

Hasil

```
MariaDB [rental_adels]> SELECT pemilik,  
-> COUNT(id_mobil) AS jumlah_mobil  
-> FROM mobil GROUP BY pemilik;  
+-----+-----+  
| pemilik | jumlah_mobil |  
+-----+-----+  
| Baim    | 1             |  
| Ibe     | 3             |  
| Ibrahim | 2             |  
| Valen   | 1             |  
+-----+-----+  
4 rows in set (0.001 sec)
```

Analisis

- `SELECT` merupakan perintah SQL yang digunakan untuk memilih data dari database.
- `pemilik` adalah nama kolom yang akan diambil dari tabel `data_mobil`.
- `COUNT(id_mobil)` adalah fungsi yang digunakan untuk menghitung jumlah baris dalam kolom `id_mobil`.
- `AS jumlah_mobil` memberikan alias pada hasil perhitungan `COUNT(id_mobil)` sehingga hasilnya akan diberi nama `jumlah_mobil`.
- `FROM data_mobil` menentukan tabel `data_mobil` sebagai sumber data.
- `GROUP BY pemilik` mengelompokkan data berdasarkan kolom `pemilik` dan melakukan perhitungan `COUNT` untuk setiap kelompok.

Kesimpulan

Perintah SQL ini akan menghasilkan daftar pemilik mobil beserta jumlah mobil yang dimiliki oleh masing-masing pemilik. Hasil query akan menampilkan dua kolom: `pemilik` yang berisi nama pemilik, dan `jumlah_mobil` yang berisi jumlah mobil yang dimiliki oleh pemilik tersebut. Perintah `GROUP BY` memastikan bahwa perhitungan `COUNT(id_mobil)` dilakukan untuk setiap pemilik secara terpisah.

4.berdasarkan query yang ada pada praktikum 5 bagian 7 tampilkan data pada table mobil dengan mengelompokkan berdasarkan pemiliknya.hitung menggunakan sum total pendapatan pemilik berdasarkan harga rental

Struktur Query

```
select data 3,SUM(data 5) AS nama_sementara from nama_tabel GROUP BY data 3;
```

Query

```
select pemilik,SUM(harga_rental) AS jumlah_pendapatan from data_mobil GROUP BY pemilik;
```

Hasil

```
MariaDB [rental_adels]> select pemilik,SUM(harga_rental) AS jumlah_pendapatan from mobil GROUP BY pemilik;
+-----+-----+
| pemilik | jumlah_pendapatan |
+-----+-----+
| Ba'im   | 50000              |
| Ibe     | 550000             |
| Ibrahim | 150000             |
| Valen   | 250000             |
+-----+-----+
4 rows in set (0.106 sec)
```

Analisis

- **SELECT** merupakan perintah yang digunakan untuk memilih data dari database.
- **pemilik** adalah nama kolom yang akan diambil dari tabel `data_mobil`.
- **SUM(harga_rental)** adalah fungsi yang digunakan untuk menghitung total nilai dari kolom `harga_rental`.
- **AS jumlah_pendapatan** memberikan alias pada hasil perhitungan `SUM(harga_rental)` sehingga hasilnya akan diberi nama `jumlah_pendapatan`.
- **FROM data_mobil** menentukan tabel `data_mobil` sebagai sumber data.
- **GROUP BY pemilik** mengelompokkan data berdasarkan kolom `pemilik` dan melakukan perhitungan `SUM` untuk setiap kelompok.

Kesimpulan

Perintah SQL ini akan menghasilkan daftar pemilik mobil beserta total pendapatan dari harga rental yang mereka miliki. Hasil query akan menampilkan dua kolom: `pemilik` yang berisi nama pemilik, dan `jumlah_pendapatan` yang berisi total pendapatan dari harga rental mobil untuk setiap pemilik.

5. Berdasarkan praktikum 5 query no 8 tampilkan jumlah pemasukan pemilik berdasarkan harga rental kelompokkan berdasarkan pemiliknya dan seleksi yang total pemasukannya atau harga rentalnya mencapai lebih besar atau sama dengan 300k

Struktur Query

```
select data_mobil, SUM(data_mobil) AS nama_sementara from nama_tabel GROUP BY
data_mobil HAVING SUM(data_mobil) >= 300000;
```

Query

```
select pemilik, SUM(harga_rental) AS jumlah_pemasukan from data_mobil GROUP BY
pemilik HAVING SUM(harga_rental) >= 300000;
```

Hasil

```
MariaDB [rental_adels]> select pemilik, SUM(harga_rental) AS jumlah_pemasukan from mobil GROUP BY pemilik HAVING SUM(harga_rental) >= 300000;
+-----+-----+
| pemilik | jumlah_pemasukan |
+-----+-----+
| Ibe     | 550000           |
+-----+-----+
1 row in set (0.037 sec)
```

Analisis

- **SELECT** merupakan perintah yang digunakan untuk memilih data dari database.
- **pemilik** adalah nama kolom yang akan diambil dari tabel `data_mobil`.
- **SUM(harga_rental)** adalah fungsi yang digunakan untuk menghitung total nilai dari kolom `harga_rental`.
- **AS jumlah_pemasukan** memberikan alias pada hasil perhitungan `SUM(harga_rental)` sehingga hasilnya akan diberi nama `jumlah_pemasukan`.
- **FROM data_mobil** menentukan tabel `data_mobil` sebagai sumber data.
- **GROUP BY pemilik** mengelompokkan data berdasarkan kolom `pemilik` dan melakukan perhitungan `SUM` untuk setiap kelompok.

- **HAVING SUM(harga_rental) >= 300000** merupakan klausa yang digunakan untuk menyaring kelompok hasil perhitungan `SUM(harga_rental)` yang nilainya lebih besar atau sama dengan 300000.

Kesimpulan

Perintah SQL ini akan menghasilkan daftar pemilik mobil beserta total pendapatan dari harga rental yang mereka miliki, tetapi hanya untuk pemilik yang total pendapatannya sama dengan atau lebih dari 300000. Hasil query akan menampilkan dua kolom: `pemilik` yang berisi nama pemilik, dan `jumlah_pemasukan` yang berisi total pendapatan dari harga rental mobil untuk setiap pemilik yang memenuhi kriteria `HAVING` tersebut.

6. Berdasarkan praktikum 6 no 12 tampilkan rata rata pemasukan pemilik mobil kelompokkan berdasarkan pemiliknya

Struktur Query

```
select nama_data,AVG(nama_data) AS nama_sementara from nama_tabel GROUP BY
nama_data;
```

Query

```
select pemilik,AVG(harga_rental) AS rata_pemasukam from data_mobil GROUP BY pemilik;
```

Hasil

```
MariaDB [rental_adels]> select pemilik,AVG(harga_rental) AS rata_pemasukam from mobil GROUP BY pemilik;
```

pemilik	rata_pemasukam
Baim	50000.0000
Ibe	183333.3333
Ibrahim	75000.0000
Valen	250000.0000

```
4 rows in set (0.040 sec)
```

Analisis

- **SELECT** merupakan perintah yang digunakan untuk memilih data dari database.
- **pemilik** adalah nama kolom yang akan diambil dari tabel `data_mobil`. Kolom ini menyimpan informasi tentang pemilik mobil.
- **AVG(harga_rental)** adalah fungsi yang digunakan untuk menghitung nilai rata-rata dari kolom `harga_rental`.

- **AS rata_pemasukan** memberikan alias pada hasil perhitungan `AVG(harga_rental)` sehingga hasilnya akan diberi nama `rata_pemasukan`.
- **FROM data_mobil** menentukan tabel `data_mobil` sebagai sumber data.
- **GROUP BY pemilik** mengelompokkan data berdasarkan kolom `pemilik` dan melakukan perhitungan `AVG` untuk setiap kelompok.

Kesimpulan

Perintah SQL ini akan menghasilkan daftar pemilik mobil beserta nilai rata-rata pendapatan dari harga rental yang mereka miliki. Hasil query akan menampilkan dua kolom: `pemilik` yang berisi nama pemilik, dan `rata_pemasukan` yang berisi rata-rata pendapatan dari harga rental mobil untuk setiap pemilik.

7. Berdasarkan praktikum 5 no 16 tampilkan pemasukan terbesar dan pemasukan terkecil kelompokkan berdasarkan pemiliknya dan seleksi data pemilik yg tampil atau memiliki jumlah mobil lebih besar dari 1.

Struktur Query

```
select nama_data, MAX(nama_data) AS nama_sementara, MIN(nama_data) AS nama_sementara
from nama_tabel GROUP BY nama_data HAVING COUNT(nama_data) >= 1;
```

Query

```
select pemilik, MAX(harga_rental) AS Pemasukan_Terbesar, MIN(harga_rental) AS
pemasukan_terkecil from data_mobil GROUP BY pemilik HAVING COUNT(harga_rental) > 1;
```

Hasil

```
MariaDB [rental_adels]> select pemilik, MAX(harga_rental) AS Pemasukan_Terbesar, MIN(harga_rental) AS pemasukan_terkecil from mobil GROUP BY pemilik
HAVING COUNT(harga_rental) > 1;
+-----+-----+-----+
| pemilik | Pemasukan_Terbesar | pemasukan_terkecil |
+-----+-----+-----+
| Ibe     | 300000             | 100000             |
| Ibrahim | 100000             | 50000              |
+-----+-----+-----+
2 rows in set (0.096 sec)
```

Analisis

- **SELECT** merupakan perintah yang digunakan untuk memilih data dari database.
- **pemilik** adalah nama kolom yang akan diambil dari tabel `data_mobil`. Kolom ini menyimpan informasi tentang pemilik mobil.
- **MAX(harga_rental)** adalah fungsi yang digunakan untuk menghitung nilai maksimum dari kolom `harga_rental`.

- **AS Pemasukan_Terbesar** memberikan alias pada hasil perhitungan `MAX(harga_rental)` sehingga hasilnya akan diberi nama `Pemasukan_Terbesar`.
- **MIN(harga_rental)** adalah fungsi yang digunakan untuk menghitung nilai minimum dari kolom `harga_rental`.
- **AS pemasukan_terkecil** memberikan alias pada hasil perhitungan `MIN(harga_rental)` sehingga hasilnya akan diberi nama `pemasukan_terkecil`.
- **FROM data_mobil** menentukan tabel `data_mobil` sebagai sumber data.
- **GROUP BY pemilik** mengelompokkan data berdasarkan kolom `pemilik` dan melakukan perhitungan `MAX` dan `MIN` untuk setiap kelompok.
- **HAVING COUNT(harga_rental) > 1** merupakan klausa yang digunakan untuk menyaring kelompok yang memiliki lebih dari satu baris data di kolom `harga_rental`.

Kesimpulan

Perintah SQL ini akan menghasilkan daftar pemilik mobil beserta nilai pemasukan terbesar dan pemasukan terkecil dari harga rental yang mereka miliki. Hasil query akan menampilkan tiga kolom: `pemilik` yang berisi nama pemilik, `Pemasukan_Terbesar` yang berisi nilai tertinggi dari harga rental, dan `pemasukan_terkecil` yang berisi nilai terendah dari harga rental untuk setiap pemilik yang memiliki lebih dari satu data rental.